Titre: Title:	Calculs Monte Carlo en transport d'énergie pour le calcul de la dose en radiothérapie sur plateforme graphique hautement parallèle
Auteur: Author:	Sami Hissoiny
Date:	2011
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Hissoiny, S. (2011). Calculs Monte Carlo en transport d'énergie pour le calcul de la dose en radiothérapie sur plateforme graphique hautement parallèle [Thèse de doctorat, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/718/

UNIVERSITÉ DE MONTRÉAL

CALCULS MONTE CARLO EN TRANSPORT D'ÉNERGIE POUR LE CALCUL DE LA DOSE EN RADIOTHÉRAPIE SUR PLATEFORME GRAPHIQUE HAUTEMENT PARALLÈLE

SAMI HISSOINY
DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR
(GÉNIE INFORMATIQUE)
DÉCEMBRE 2011

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée:

CALCULS MONTE CARLO EN TRANSPORT D'ÉNERGIE POUR LE CALCUL DE LA DOSE EN RADIOTHÉRAPIE SUR PLATEFORME GRAPHIQUE HAUTEMENT PARALLÈLE

présentée par: HISSOINY Sami
en vue de l'obtention du diplôme de: Philosophiæ Doctor
a été dûment acceptée par le jury d'examen constitué de:

M. DAGENAIS Michel, Ph.D., président
M. OZELL Benoît, Ph.D., membre et directeur de recherche
M. DESPRÉS Philippe, Ph.D., membre et codirecteur de recherche
M. HÉBERT Alain, D. Ing., membre
M. ARCHAMBAULT Louis, Ph.D., membre

REMERCIEMENTS

À Philippe et Benoît, superviseurs et collaborateurs de longue date.

À Bas, qui a su m'accueillir et me montrer le charme des Pays-Bas.

À Hugo, pour les prouesses mathématiques et les discussions.

À Marie-Pier, qui a enduré des soirées et des matins de fin de semaine de travail.

À Christopher, qui a relu les articles de cette thèse et qui sait maintenant que Compton est fantastique.

À mes parents qui m'ont permis de manger malgré tout.

RÉSUMÉ

Le calcul de dose est au centre de la planification de traitement. Ce calcul de dose doit être 1) assez précis pour que le physicien médical et le radio-oncologue puissent prendre une décision basée sur des résultats près de la réalité et 2) assez rapide pour permettre une utilisation routinière du calcul de dose. Le compromis entre ces deux facteurs en opposition a donné lieu à la création de plusieurs algorithmes de calcul de dose, des plus approximatifs et rapides aux plus exacts et lents. Le plus exact de ces algorithmes est la méthode de Monte Carlo puisqu'il se base sur des principes physiques fondamentaux.

Depuis 2007, une nouvelle plateforme de calcul gagne de la popularité dans la communauté du calcul scientifique : le processeur graphique (GPU). La plateforme existe depuis plus longtemps que 2007 et certains calculs scientifiques étaient déjà effectués sur le GPU. L'année 2007 marque par contre l'arrivée du langage de programmation CUDA qui permet de faire abstraction du contexte graphique pour programmer le GPU. Le GPU est une plateforme de calcul massivement parallèle et adaptée aux algorithmes avec données parallèles.

Cette thèse vise à savoir comment maximiser l'utilisation d'une plateforme GPU en vue d'améliorer la vitesse d'exécution de la simulation de Monte Carlo en transport d'énergie pour le calcul de la dose en radiothérapie.

Pour répondre à cette question, la plateforme GPUMCD a été développée. GPUMCD implémente une simulation de Monte Carlo couplée photon-électron et s'exécute complètement sur le GPU.

Le premier objectif de cette thèse est d'évaluer cette méthode pour un calcul en radiothérapie externe. Des sources monoénergétiques simples et des fantômes en couches sont utilisés. Une comparaison aux plateforme EGSnrc et DPM est effectuée. GPUMCD reste à l'intérieur de critères gamma 2%-2mm de EGSnrc tout en étant au moins 1200× plus rapide qu'EGSnrc et 250× plus rapide que DPM.

Le deuxième objectif consiste en l'évaluation de la plateforme pour un calcul en curiethérapie interne. Des sources complexes basées sur la géométrie et le spectre énergétique de sources réelles sont utilisées à l'intérieur d'une géométrie de type TG-43. Des différences de moins de 4% sont trouvées lors d'une comparaison à la plateforme BrachyDose ainsi qu'aux données

consensus du TG-43.

Le troisième objectif vise l'utilisation de GPUMCD comme engin de calcul pour le MRI-Linac. Pour ce faire, la prise en considération de l'effet du champ magnétique sur les particules doit être ajoutée. Il a été démontré que GPUMCD se situe à l'intérieur de critères gamma 2%-2mm de deux expériences visant à mettre en évidence l'influence du champ magnétique sur les distributions de dose.

Les résultats suggèrent que le GPU est une plateforme matérielle intéressante pour le calcul de dose par simulation de Monte Carlo et que la plateforme logicielle GPUMCD permet de faire un calcul rapide et exact.

ABSTRACT

Dose calculation is a central part of treatment planning. The dose calculation must be 1) accurate so that the medical physicists and the radio-oncologists can make a decision based on results close to reality and 2) fast enough to allow a routine use of dose calculation. The compromise between these two factors in opposition gave way to the creation of several dose calculation algorithms, from the most approximate and fast to the most accurate and slow. The most accurate of these algorithms is the Monte Carlo method, since it is based on basic physical principles.

Since 2007, a new computing platform gains popularity in the scientific computing community: the graphics processor unit (GPU). The hardware platform exists since before 2007 and certain scientific computations were already carried out on the GPU. Year 2007, on the other hand, marks the arrival of the CUDA programming language which makes it possible to disregard graphic contexts to program the GPU. The GPU is a massively parallel computing platform and is adapted to data parallel algorithms.

This thesis aims at knowing how to maximize the use of a graphics processing unit (GPU) to speed up the execution of a Monte Carlo simulation for radiotherapy dose calculation.

To answer this question, the GPUMCD platform was developed. GPUMCD implements the simulation of a coupled photon-electron Monte Carlo simulation and is carried out completely on the GPU.

The first objective of this thesis is to evaluate this method for a calculation in external radiotherapy. Simple monoenergetic sources and phantoms in layers are used. A comparison with the EGSnrc platform and DPM is carried out. GPUMCD is within a gamma criteria of 2%-2mm against EGSnrc while being at least 1200× faster than EGSnrc and 250× faster than DPM.

The second objective consists in the evaluation of the platform for brachytherapy calculation. Complex sources based on the geometry and the energy spectrum of real sources are used inside a TG-43 reference geometry. Differences of less than 4% are found compared to the BrachyDose platforms well as TG-43 consensus data.

The third objective aims at the use of GPUMCD for dose calculation within MRI-Linac

environment. To this end, the effect of the magnetic field on charged particles has been added to the simulation. It was shown that GPUMCD is within a gamma criteria of 2%-2mm of two experiments aiming at highlighting the influence of the magnetic field on the dose distribution. The results suggest that the GPU is an interesting computing platform for dose calculations through Monte Carlo simulations and that software platform GPUMCD makes it possible to achieve fast and accurate results.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES NOTATIONS ET DES SYMBOLES

GPU	: *graphics processing unit*, unité de traitement graphique
pdf	: densité de probabilité
CDF	: distribution cumulative de probabilité
MFP	: distance moyenne à la prochaine interaction
E	: énergie $[eV]$
u	: unité de masse atomique $(1.6605402 \times 10^{-24} g)$
μ	: moyenne statistique
$\mu(E)$	: coefficient d'atténuation $[cm^{-1}]$ ou $[cm^{-2} \cdot g^{-1}]$
$\bar{X}$	: moyenne échantillonnée
σ	: section efficace
A	: masse atomique
N_A	: nombre d'Avogadro $(6.022 \times 10^{23}\ mol^{-1})$
σ	: section efficace $[cm^{-1}]$
D	: dose $[J \cdot Kg^{-1}]$
c	: vitesse de la lumière $(\approx 3 \times 10^8\ m/s)$
$m_o c^2$	: énergie de l'électron au repos, 511 keV

INDICES

ray	: diffusion de Rayleigh
$comp$	: diffusion de Compton
$photo$	: effet photoélectrique
$pair$	: production de paires
$inel$	: collision inélastique
el	: collision élastique
$brem$	: production de *bremsstrahlung*
e	: électron
p	: photon
$SIMD$	: Instruction Simple Données Multiples (*Single Instruction Multiple Data*)

DPM	: Dose Planning Method, un logiciel Monte Carlo rapide
$TG-43$	: Task Group 43, rapport sur le calcul de dose en curiethérapie
MLC	: Collimateur Multi-Lames (*Multi-Leaf Collimator*)
$IMRT$	: Radio-Thérapie Modulée en Intensité (*Intensity Modulated RatioTherapy*)

LISTE DES ANNEXES

INTRODUCTION

Le traitement de radiothérapie est l'un des traitements possibles pour tenter l'élimination des tissus cancéreux. La méthodologie consiste à diriger de la radiation ionisante, le plus souvent émise par un accélérateur linéaire (*linac*), vers les tissus cancéreux tout en évitant le plus possible les tissus et organes sains.

Au coeur de la planification, étape nécessaire avant que le traitement ait lieu, se trouve le calcul de la dose reçue. Le planification de traitement vise à trouver la meilleure configuration de faisceaux ionisants pour répondre à la prescription de la dose aux tissus, telle qu'élaborée par le radio-oncologue. Ce calcul fait intervenir les principes physiques du transport de l'énergie dans la matière ainsi que les informations sur la composition du milieu préalablement obtenues par imagerie médicale. À l'étape du calcul de la dose, plusieurs algorithmes peuvent être utilisés, des plus rapides, faisant plusieurs approximations quant à la nature du milieu et aux interactions particules-matière, aux plus exacts qui font le compromis d'un long temps de calcul pour une fidélité augmentée aux principes physiques et au milieu. Dans la famille des techniques rapides, nous retrouvons les méthodes couramment utilisées du faisceau pinceau et de la convolution superposition. Pour les méthodes les plus exactes, mentionnons les solutions stochastiques (simulation Monte Carlo) et déterministes des équations de Boltzmann couplées. Des différences de plus de 10% peuvent être observées lors de la comparaison entre algorithmes exacts et rapides (Krieger and Sauer, 2005; Ding et al., 2005).

Par ailleurs, l'intégration du collimateur multilames a permise de mettre en forme la fluence en sortie de l'accélérateur linéaire pour que celle-ci se colle le plus possible à la cible visée. Le problème du calcul de la dose est alors habituellement défini comme un problème inverse : « Quelle configuration de faisceaux donne la dose souhaitée au volume cible et épargne les tissus sains ? ». Pour résoudre ce problème inverse, la dose doit être calculée plusieurs fois à l'intérieur d'un méta-algorithme d'optimisation de contraintes. Pour accélérer le calcul de la solution, un algorithme de calcul de dose rapide est habituellement utilisé lors des phases d'optimisation et un calcul plus exact est utilisé pour vérifier la solution finale. Idéalement,

le calcul exact serait utilisé tout au long du calcul de la solution optimale pour effectivement s'assurer qu'elle est optimale. Des différences importantes de plus de 10% ont encore une fois été trouvées en comparant la solution par algorithme rapide à la solution par algorithme exact (Ma et al., 2000).

Dans un autre ordre d'idée, l'informatique axée sur le calcul scientifique a vu apparaître au cours des dernières années une nouvelle plateforme matérielle avec des caractéristiques intéressantes pour le type de calcul habituellement rencontré dans ce milieu : le processeur graphique (GPU, *graphics processing unit*). Le GPU est un processeur massivement parallèle, visant un grand nombre d'unités de calcul au détriment d'une petite cache de données. Cette plateforme n'est donc pas idéale pour tous les types de problèmes, mais ceux retrouvés en calcul scientifique étant souvent de nature parallèle, l'intérêt pour la plateforme est grandissante,par exemple (Satish et al., 2009; Micikevicius, 2009; Volkov and Demmel, 2008; Thomas et al., 2009; Wang et al., 2009). Le GPU a aussi trouvé une niche dans le domaine de la physique médicale, par exemple (de Greef et al., 2009; Kutter et al., 2009a; Kutter et al., 2009b; Badal and Badano, 2009; Després et al., 2008).

Le but de cette thèse est l'étude de l'utilisation du processeur graphique (GPU) pour l'implémentation d'une plateforme de calcul de dose par simulation de Monte Carlo dans un contexte de calculs en radiothérapie. La solution GPU n'existant pas à ce jour, elle devra être développée.

Le corps de cette thèse est présenté sous la forme de trois articles scientifiques. Ces articles ont pour objectif de présenter et de valider, en termes de rapidité et d'exactitude, la plateforme dans trois contexte différents : radiothérapie externe avec et sans champ magnétique et curiethérapie interne.

Ce document est divisé de la façon suivante : le chapitre 1 présente la théorie nécessaire et effectue une revue critique de la littérature. Le chapitre 2 décrit la démarche de l'ensemble du travail et l'organisation générale du document. Le chapitre 3 présente la description et la validation de la plateforme GPUMCD dans un cadre de radiothérapie externe. Le chapitre 4 présente les modifications à la plateforme GPUMCD ainsi que sa validation dans un cadre de

curiethérapie. Le chapitre 5 présente les modifications à la plateforme GPUMCD ainsi que sa validation pour une calcul en radiothérapie externe en présence d'un fort champ magnétique pour une utilisation de calcul de dose pour le MRI-Linac (Lagendijk et al., 2008). Le chapitre 6 présente un ensemble d'aspects méthodologiques ne figurant dans aucun article. Finalement, le chapitre 7 propose une discussion générale sur le projet, son impact et ses ouvertures.

CHAPITRE 1

REVUE DE LITTÉRATURE EN CALCUL DE DOSE MONTE CARLO ET SIMULATIONS SUR GPU

Ce chapitre s'attarde aux concepts et à la littérature pertinente reliés au calcul de dose par Monte Carlo, tant du point de vue physique qu'informatique. La section. 1.1 présente le problème du transport d'énergie, l'équation de Boltzmann, et des algorithmes autres que la méthode de Monte Carlo pour la résoudre. La section 1.2 décrit la méthode de Monte Carlo et son application au problème de transport d'énergie. La section 1.3 détaille des méthodes existantes pour faire du transport d'énergie par simulation Monte Carlo. La section 1.4 décrit la plateforme matérielle, le GPU, ainsi que son environnement de programmation. La section 1.5 énumère des applications récentes du GPU à fins de simulation. Finalement, la section. 1.6 décrit le but et les objectifs de cette étude.

1.1 Transport d'énergie et autres algorithmes de calcul de dose

Cette section traitera ultimement de l'équation de Boltzmann ainsi que des diverses façons de la résoudre. Il est par contre nécessaire de tout d'abord introduire les notions de sections efficaces, de pouvoir d'arrêt et d'interactions entre particules, ce qui fait l'objet de la section 1.1.1. Par la suite, en section 1.1.2, l'équation de Boltzmann est détaillée. En section 1.1.3, différents algorithmes de résolution de l'équation de Boltzmann, et donc de calcul de dose, sont présentés.

1.1.1 Sections efficaces, pouvoir d'arrêt et interactions entre particules

Les particules énergétiques, chargées ou non, interagissent avec la matière de différentes manières. Ces interactions font que l'énergie initiale de la particule énergétique est ultimement

déposée dans le milieu, ce qui correspond à la quantité d'intérêt de ce projet, la dose. La dose se définit comme l'énergie déposée par unité de masse :

$$D = \frac{E}{m}\left[\frac{J}{kg}\right]. \tag{1.1}$$

1.1.1.1 Sections efficaces

Centrale à la description des différentes interactions se trouve le concept de section efficace.

Dans un sens général, la section efficace décrit la probabilité, par unité de longueur, qu'une particule interagisse avec une des particules formant le milieu par unité de volume :

$$\sigma = \frac{dp}{Ndz}, \tag{1.2}$$

où σ est la section efficace, p la probabilité d'un événement quelconque, N la densité atomique et z la distance parcourue. Il est plus habituel d'approcher le problème de transport d'énergie en fonction de sections efficaces différentielles et totales. La section efficace différentielle définit la distribution de probabilité d'interaction en fonction de l'énergie et de l'orientation, avant et après interaction, des différentes particules impliquées. L'expression de section efficace, σ, se réécrit donc, sous sa forme différentielle, comme :

$$\frac{d^2\sigma}{dEd\Omega}(x, E', E, \Omega', \Omega)dEd\Omega\left[\frac{1}{cm \cdot MeV \cdot sr}\right]. \tag{1.3}$$

Cette expression représente la probabilité par unité de longueur qu'une particule avec énergie E' et orientation Ω' à un point x soit diffusée avec une énergie de E et une orientation autour de Ω. La section efficace totale quant à elle correspond à la section efficace différentielle intégrée pour toutes les énergies et orientations :

$$\sigma(E) = \int dE' \int_{4\pi} \sigma(E, E', \Omega, \Omega')d\Omega. \tag{1.4}$$

À partir de cette définition, il est possible d'établir le concept de coefficient d'atténuation,

μ, qui correspond au produit de la section efficace totale et du nombre de particule dans le milieu, n.

$$\mu(E) = n\sigma(E) = n \int dE' \int_{4\pi} \sigma(E, E', \Omega, \Omega') d\Omega \left[\frac{1}{cm}\right], \tag{1.5}$$

avec $n = \frac{\rho}{A} N_A$ où ρ est la densité, A la masse atomique et N_A le nombre d'Avogadro. Les données tabulées du coefficient d'atténuation¸, par exemple (Berger et al., 1998), sont habituellement normalisées par la densité pour ne pas laisser cette quantité dépendante du milieu. Les unités de μ deviennent donc $[cm^{-2} \cdot g^{-1}]$.

1.1.1.2 Pouvoir d'arrêt

Un concept corollaire à la notion de section efficace est la notion de pouvoir d'arrêt pour les particules chargées. Il se définit comme étant la vitesse à laquelle une particule perd de l'énergie à mesure qu'elle progresse dans un milieu. Mathématiquement :

$$S(E) = -\frac{dE}{dx} \left[\frac{eV}{cm}\right], \tag{1.6}$$

où $S(E)$ est le pouvoir d'arrêt. Ce concept est lié à celui de section efficace :

$$S(x, E') = \int_0^\infty (E' - E) \frac{d\sigma}{dE}(E', E) dE. \tag{1.7}$$

Le pouvoir d'arrêt inclut donc, indifféremment, toutes les interactions possibles entre particule chargée et matière.

1.1.1.3 Interactions entre particules et matière

1.1.1.3.1 Photons Cette section porte sur les interactions possibles entre photons et matière. Dans le domaine d'énergie de la physique médicale, ces interactions sont au nombre de quatre mais les collisions élastiques (diffusion de Rayleigh) ne seront pas explicitées de par leur contribution négligeable à la dose ; reste la collision inélastique (diffusion de Compton), l'effet photoélectrique et la production de paire. Nous pouvons donc définir le coefficient

d'atténuation total, pour les photons, comme étant :

$$\mu_p(E) = \mu_{comp}(E) + \mu_{photo}(E) + \mu_{pair}(E). \tag{1.8}$$

Les interactions entre photons et matière ne déposent pas directement d'énergie et les photons ne sont donc pas directement responsables du dépôt de dose. Par contre, les photons mettent en mouvement des électrons qui, eux, déposent de la dose. Les équations de sections efficaces, lorsqu'elles sont simples, seront présentées pour que la discussion soit complète. Elles sont tirées directement des documents de référence.

Lors d'une interaction de type diffusion Compton, un photon avec une énergie E_p et une direction Ω_p entre en collision avec un électron et lui transmet une partie de sa quantité de mouvement. À la suite de cette interaction se retrouvent un photon dévié et avec une énergie réduite E'_p et direction Ω'_p et un électron avec énergie E'_e et direction Ω'_e. Les électrons rencontrés dans un problème de radiothérapie ne sont pas libres. Une partie de l'énergie doit donc être utilisée pour extraire l'électron de son atome. De plus, l'atome retournera à son état fondamental et un rayonnement caractéristique sera émis. La conservation d'énergie s'écrit donc comme $E_p = E'_p + E'_e + I$ où I est l'énergie d'ionisation. Il est d'usage, à haute énergie (>1 keV), de négliger ces deux facteurs pour calculer le coefficient d'atténuation pour l'interaction de type Compton qui peut alors se trouver à l'aide de l'équation de Klein-Nishina (Klein and Nishina, 1929) :

$$\frac{d^2\sigma_{comp}}{d\Omega' dE'} = C\left(\frac{E'}{E}\right)^2\left(\frac{E'}{E} + \frac{E}{E'} - sin^2(\theta)\right), \tag{1.9}$$

où θ est l'angle entre le photon incident et diffusé (c'est-à-dire $cos(\theta) = \Omega \cdot \Omega'$), E' l'énergie du photon diffusé, et C une constante de proportionnalité.

Lors d'une interaction de type photoélectrique, un photon incident avec énergie E_p et orientation Ω_p donne toute son énergie à un électron. À la suite de cette interaction le photon a donc été absorbé et un nouvel électron possédant une énergie $E'_e = E_p - I$, où I est encore une fois l'énergie d'ionisation, et une orientation Ω'_e est mis en mouvement. La section efficace

différentielle pour l'orientation d'un électron de couche K est donnée par la distribution de Sauter (Sauter, 1931) :

$$\frac{d\sigma_{photo}}{d\Omega'_e} = C\frac{\beta^3}{\gamma}\frac{sin^2\theta_e}{(1-\beta cos\theta_e)^2}\left[1+\frac{1}{2}\gamma(\gamma-1)(\gamma-2)(1-\beta cos\theta_e)\right] \tag{1.10}$$

où C est une constante de proportionnalité, $cos(\theta_e) = \Omega_P \cdot \Omega'_e$, β est la vitesse de l'électron en unité de vitesse lumière et $\gamma = \frac{1}{\sqrt{1-\beta^2}}$.

Lors d'une interaction de type production de paire, un photon avec une énergie E_p est absorbé et une paire électron/positron est créée. L'énergie minimale du photon incident est donc $E_p \geq 2m_0c^2$ où m_0 est la masse de l'électron au repos et c la vitesse de la lumière. L'équation de la section efficace ne sera pas présentée ici puisqu'elle dépend de plusieurs termes complexes. Elle peut être trouvée à l'Eq. 2.77 du document sur PENELOPE (Salvat et al., 2003).

1.1.1.3.2 Électrons Les types d'interactions entre électrons et matière sont au nombre de trois : collision élastique, collision inélastique et production de *bremsstrahlung*. Le coefficient d'atténuation total pour les électrons s'écrit donc :

$$\mu_e(E) = \mu_{inel}(E) + \mu_{el}(E) + \mu_{brem}(E) \tag{1.11}$$

Lors d'une collision inélastique, un électron énergétique incident avec énergie E_{e1} et orientation Ω_{e1} entre en collision avec un électron assumé au repos. Suite à cette interaction, un électron a une énergie E'_{e1} et une orientation Ω'_{e1} et l'autre a une énergie E'_{e2} et une orientation Ω'_{e2}. De part le principe d'indissociabilité des électrons, l'électron avec l'énergie la plus grande est considéré comme l'électron diffusé (celui avec l'indice '1'). La conservation d'énergie donne $E_{e1} = E'_{e1} + E'_{e2} + I$. Si l'énergie d'ionisation est ignorée, l'interaction peut être décrite par la section efficace de Møller (Møller, 1932) :

$$\frac{d\sigma_{inel}}{dE'_{e2}} = \frac{C}{\beta^2}\left(\frac{1}{E'_{e2}} + \frac{1}{(E_{e1}-E'_{e2})^2} + \frac{\epsilon^2}{(\epsilon+1^2E^2_{e1}} - \frac{2\epsilon+1}{(\epsilon+1)^2E'_{e2}(E_{e1}-E'_{e2})}\right), \tag{1.12}$$

où $\epsilon = \frac{E_{e1}}{m_0 c^2}$, $\beta^2 = 1 - (1+\epsilon)^2$ et C est une constante de proportionnalité. La section différentielle l'est seulement par rapport à l'énergie de l'électron secondaire ; l'orientation des deux électrons est définie par la cinématique et n'est donc pas une quantité aléatoire.

Lors d'une collision élastique, un électron énergétique incident avec énergie E_e et orientation Ω_e conserve son énergie mais est dévié avec un angle $\theta = \Omega_e \cdot \Omega'_e$. Ce mécanisme ne contribue donc pas au dépôt de dose mais, vu le nombre élevé de collisions élastiques pour chaque électron, il s'agit d'un mécanisme important de déviation de trajectoire. L'interaction est définie par la section efficace différentielle de Rutherford (Kawrakow and Rogers, 2000) :

$$\frac{d\sigma_{el}}{d\mu} = \frac{C}{\beta^2 \tau(\tau+2)} \frac{1}{(1-\mu+2\eta)^2}, \tag{1.13}$$

où β est la vitesse de la particule en unité de vitesse lumière, τ l'énergie cinétique en unité de m et $\mu = cos(\theta)$. η est un paramètre d'effet d'écran qui peut se développer de diverses façons qui ne seront pas détaillées ici. Une discussion est disponible à la référence sus-mentionnée.

Lors d'une interaction de type *bremsstrahlung*, un électron incident d'énergie E_e et orientation Ω_e est ralenti et dévié à une énergie E'_e et orientation Ω'_e et un photon énergétique est créé avec énergie E'_p et orientation Ω'_p. Le développement de l'expression de la section efficace est, similairement à l'interaction de production de paires, omis de la discussion. Une discussion est disponible dans (Kawrakow and Rogers, 2000) et le travail fondateur est disponible dans (Koch and Motz, 1959).

1.1.2 Équation de Boltzmann

L'équation de transport Boltzmann peut être utilisée pour décrire l'évolution de particules dans le milieu. Elle est à la base, de près ou de loin, de tous les algorithmes de calcul de dose modernes. L'équation de Boltzmann est une équation d'équilibre. Elle égale le nombre de particules dans un élément de volume ΔV autour d'un point $\vec{x}$, en fonction de leur énergie E et de leur quantité de mouvement normalisée $\vec{\Omega}$; avec le nombre de particules créé dans le volume ainsi que la somme des interactions ayant eu lieu dans le volume et les particules

quittant le volume. Une dérivation rapide de l'équation de Boltzmann est présentée dans cette thèse par souci de complétude alors qu'une dérivation plus générale peut être trouvée dans (Bouchard, 2010; Leppanen, 2007).

Quelques définitions préalables sont nécessaires pour arriver à la formulation de la dose. Tout d'abord, $\eta(\vec{x}, \vec{\Omega}, E, t)$ la densité de particule. Par définition nous trouvons que $\eta(\vec{x}, \vec{\Omega}, E, t)d\vec{x}d\vec{\Omega}dEdt$ est le nombre de particules avec énergie E autour de dE et direction $\vec{\Omega}$ autour de $d\vec{\Omega}$ à la position $\vec{x}$ dans $d\vec{x}$ au temps t autour de dt.

À partir de la définition générale de l'équation de Boltzmann, qui balance ce qui entre, ce qui sort et ce qui est produit à l'intérieur d'une unité de volume, nous posons l'équation suivante :

$$\frac{d\eta(\vec{x}, \vec{\Omega}, E, t)}{dt} = \text{Particules produites} + \text{Interactions de particules} \tag{1.14}$$

Le nombre de particules traversant le point $\vec{x}$ par unité de temps est défini comme :

$$\vec{j}(\vec{x}, \vec{\Omega}, E, t) = \frac{\partial \eta(\vec{x}, \vec{\Omega}, E, t)}{\partial t} \tag{1.15}$$

et le flux de particules comme :

$$\phi(\vec{x}, \vec{\Omega}, E, t) = v \cdot \eta(\vec{x}, \vec{\Omega}, E, t), \tag{1.16}$$

où v est la vitesse de la particule. Nous trouvons donc

$$\phi(\vec{x}, \vec{\Omega}, E, t) = \frac{\partial \eta(\vec{x}, \vec{\Omega}, E, t)}{\partial s} \tag{1.17}$$

où s est la longueur de trajet de la particule.

De Eq. 1.14 et Eq. 1.16, nous avons :

$$\frac{1}{v}\frac{d\phi(\vec{x}, \vec{\Omega}, E, t)}{dt} = \frac{1}{v}\frac{\partial \phi(\vec{x}, \vec{\Omega}, E, t)}{dt} + \vec{\Omega}\nabla\phi(\vec{x}, \vec{\Omega}, E, t) = S + I, \tag{1.18}$$

où S est le terme de source et I le terme d'intéraction.

En intégrant sur le temps, nous obtenons l'équation de Boltzmann indépendante du temps :

$$\vec{\Omega}\nabla\varphi(\vec{x},\vec{\Omega},E,t) = S + I, \tag{1.19}$$

où $\varphi(\vec{x},\vec{\Omega},E,t)$ est la fluence, définie comme :

$$\varphi(\vec{x},\vec{\Omega},E,t) = \int_0^\infty \phi(\vec{x},\vec{\Omega},E,t)dt. \tag{1.20}$$

À partir des définitions de la section 1.1.1, deux termes d'interactions sont définis : I_p pour les photons et I_e pour les électrons. Nous trouvons (Bouchard, 2010) :

$$\begin{aligned}
I_p(\vec{x},\vec{\Omega},E_p) = &- \int\int_{4\pi} \varphi_p(\vec{x},\vec{\Omega},E_p)\mu_{photo}(\vec{x},E_p,\vec{\Omega},E_p-I,\vec{\Omega}')d\vec{\Omega}' \\
&- \int_0^{E_p-I} dE'_p \int\int_{4\pi} \varphi_p(\vec{x},\vec{\Omega},E_p)\mu_{comp}(\vec{x},E_p,\vec{\Omega},E'_p,\vec{\Omega}')d\vec{\Omega}' \\
&- \int_0^{E_p-2m_0c^2} dE'_e \int\int_{4\pi} \varphi_p(\vec{x},\vec{\Omega},E_p)\mu_{pair}(\vec{x},E_p,\vec{\Omega},E'_e,\vec{\Omega}')d\vec{\Omega}' \\
&+ \int_0^{E_p-I} dE'_p \int\int_{4\pi} \varphi_p(\vec{x},\vec{\Omega},E')\mu_{comp}(\vec{x},E'_p,\vec{\Omega},E_p,\vec{\Omega}')d\vec{\Omega}' \\
&+ \int_0^{E_e} dE'_e \int\int_{4\pi} \varphi_e(\vec{x},\vec{\Omega},E'_e)\mu_{brem}(\vec{x},E'_e,\vec{\Omega}',E_p,\vec{\Omega})d\vec{\Omega}'
\end{aligned} \tag{1.21}$$

où les coefficients d'atténuations μ utilisés sont basés sur les sections efficaces différentielles et non totales. Le premier terme correspond à la perte de photons due à l'effet photoélectrique, le deuxième à la perte due à l'effet Compton et le troisième à la perte due à l'effet de production de paires. Le quatrième terme représente le photon dévié lors de l'effet Compton et

le cinquième le gain d'un photon lors d'une interaction de *bremsstrahlung*. De façon similaire :

$$\begin{aligned} I_e(\vec{x},\vec{\Omega},E_e) = &+ \int\int_{4\pi} \varphi_p(\vec{x},\vec{\Omega},E'_p)\mu_{photo}(\vec{x},E'_p,\vec{\Omega},E_e-I,\vec{\Omega}')d\vec{\Omega}' \\ &+ \int_0^{E_p-I} dE'_e \int\int_{4\pi} \varphi_p(\vec{x},\vec{\Omega},E'_p)\mu_{comp}(\vec{x},E'_p,\vec{\Omega},E_e,\vec{\Omega}')d\vec{\Omega}' \\ &+ \int_0^{E_p-2m_0c^2} dE'_e \int\int_{4\pi} \varphi_p(\vec{x},\vec{\Omega},E'_p)\mu_{pair}(\vec{x},E'_p,\vec{\Omega},E_e,\vec{\Omega}')d\vec{\Omega}' \\ &+ \int_0^{\frac{E'_e+I}{2}} dE'_e \int\int_{4\pi} \varphi_e(\vec{x},\vec{\Omega},E'_e)\mu_{inel}(\vec{x},E'_e,\vec{\Omega},E_e,\vec{\Omega}')d\vec{\Omega}' \\ &+ \int_{\frac{E'_e+I}{2}}^{E_e} dE'_e \int\int_{4\pi} \varphi_e(\vec{x},\vec{\Omega},E'_e)\mu_{inel}(\vec{x},E'_e,\vec{\Omega},E_e,\vec{\Omega}')d\vec{\Omega}' \\ &- \int_0^{E_e} dE'_e \int\int_{4\pi} \varphi_e(\vec{x},\vec{\Omega},E_e)\mu_{inel}(\vec{x},E_e,\vec{\Omega},E'_e,\vec{\Omega}')d\vec{\Omega}' \\ &- \int_0^{E_e} dE'_p \int\int_{4\pi} \varphi_e(\vec{x},\vec{\Omega},E_e)\mu_{brem}(\vec{x},E_e,\vec{\Omega}',E'_p,\vec{\Omega})d\vec{\Omega}' \\ &+ \int_0^{E_e} dE'_e \int\int_{4\pi} \varphi_e(\vec{x},\vec{\Omega},E'_e)\mu_{brem}(\vec{x},E'_e,\vec{\Omega}',E_e,\vec{\Omega})d\vec{\Omega}' \end{aligned} \tag{1.22}$$

où les coefficients d'atténuations différentiels sont encore utilisés. Les trois premiers termes représentent des gains d'électrons dus aux interactions entre photons et matière et les quatrième et cinquième termes les électrons résultant d'une collision inélastique. Le sixième terme représente l'électron incident lors de la collision élastique, le septième terme est responsable de la perte d'une partie de l'énergie de l'électron pour créer un photon lors de *bremsstralung* et le huitième est l'énergie restante de l'électron.

À ce point, l'équation 1.19 peut être utilisée pour calculer la dose déposée en tout point. Physiquement, la dose correspond à la somme des énergies d'ionisation libérées par unité de volume. De façon plus pratique, la notion de pouvoir d'arrêt peut être utilisée pour calculer la dose :

$$D(\vec{x})m(\vec{x}) = \int_0^{\infty} S(\vec{x},E) \int_{4\pi} \varphi(\vec{x},\Omega,E)d\Omega dE \tag{1.23}$$

1.1.3 Autres algorithmes de calcul de dose

Cette section fait état des alternatives à la simulation de Monte Carlo pour le calcul de la dose. Nous faisons abstraction des méthodes ancestrales du calcul de dose, qui se basaient très peu sur les données spécifiques aux patients, pour se concentrer sur trois méthodes modernes : le faisceau pinceau, la convolution/superposition et la méthode déterministe de résolution de l'équation de Boltzmann.

1.1.3.1 Le faisceau pinceau

Les algorithmes présentés, notamment celle du faisceau pinceau, ont leur source dans diverses approximations de l'équation de Boltzmann. Tout le détail mathématique n'est pas pertinent pour ce travail et est présenté dans un article de Börgers et Larsen (Borgers and Larsen, 1996). En termes généraux, le faisceau pinceau se base sur l'approximation de Fokker-Plank et de Fermi pour résoudre l'équation de Boltzmann. L'approximation de Fokker-Plank émet l'hypothèse que la distribution de probabilité de l'angle de diffusion d'un électron après collision favorise fortement un petit angle. L'approximation de Fermi va une étape plus loin et évalue l'équation de Fokker-Plank en ne gardant que le terme principal d'un développement de Taylor de cette dernière. Ce développement mathématique pourrait en théorie permettre une solution analytique. En pratique, il est rare que ceci se réalise et une approche par noyau (*kernel*) est plutôt employée (Ahnesjö et al., 1992; Jeleń et al., 2005). Le noyau est le résultat d'une simulation Monte Carlo où un faisceau mince interagit avec un cube d'eau et où le résultat, en termes de distribution de dose, est conservé. Par la suite, lors d'une situation de calcul de dose, le faisceau primaire émanant de la source externe est divisé en un certain nombre de faisceaux pinceaux. La dose totale est calculée comme étant la superposition de la contribution de chacun des faisceaux pinceaux composant le faisceau primaire. La subdivision du faisceau doit avoir la propriété d'auto consistance (*self consistency*) qui assure que la superposition de N faisceaux pinceaux, dans un fantôme hétérogène (où différents matériaux de propriétés d'atténuation différentes sont présents), donne la même dose que si le faisceau primaire avait été divisé en 2N ou 0.5N faisceaux pinceaux. Le faisceau pinceau peut tenir

compte des hétérogénéités en profondeur en mettant à l'échelle le noyau et en calculant une profondeur équivalente en eau.

1.1.3.2 La convolution-superposition

La méthode par convolution-superposition (CS) (Mackie et al., 1985; Liu et al., 1997) est similaire à la méthode du faisceau pinceau en ce sens que les deux utilisent des noyaux extraits de simulations de Monte Carlo. Si la méthode faisceau pinceau utilise un noyau résultant de l'interaction entre une ligne et un fantôme, la méthode de CS résulte de l'interaction entre un point et un fantôme. En effet, les particules primaires sont forcées d'interagir en un point et la simulation Monte Carlo établit comment l'énergie est diffusée autour de ce point, en prenant en considération la direction du faisceau incident. La méthode de CS se divise en deux parties : la phase de TERMA (*total energy release per mass unit*, énergie totale transferée par unité de masse) et la phase de superposition ou convolution. La phase de TERMA consiste à calculer la fluence due aux particules primaires émanant de la source externe à chaque point dans le fantôme et se fait habituellement par un traçage de rayon entre la source et le point de calcul. La phase de convolution/superposition utilise le noyau pour modéliser le diffusé de radiation en convoluant/superposant la valeur de la TERMA en un point et le noyau obtenu par simulation Monte Carlo. La prise en charge des hétérogénéités se fait aux deux étapes. Lors de la phase de TERMA, la fluence en tout point obéit à une atténuation exponentielle décroissante ainsi qu'à l'inverse du carré de la distance. Ces deux éléments peuvent être mis à l'échelle en prenant une longueur équivalente dans l'eau plutôt qu'une longueur géométrique. De même, lors de la phase de superposition (la convolution exclut le traitement des hétérogénéités puisque le noyau doit être invariant pour que le théorème de convolution s'applique), le noyau peut être mis à l'échelle pour tenir compte des hétérogénéités. Le principe de convolution/superposition peut modéliser exactement le diffusé de premier ordre c'est-à-dire les particules secondaires émises des particules primaires mais il ne peut qu'approximer le diffusé d'ordres supérieurs (Bielajew, 2001).

1.1.3.3 Solution déterministe de l'équation de Boltzmann

La méthode déterministe (BOMAN, 2007; Hensel et al., 2006) diffère des deux autres méthodes présentées précédemment. Elle peut être considérée analogue aux simulations de Monte Carlo en termes d'exactitude et de physique incluse lors du calcul, mais utilise une méthode différente pour résoudre l'équation de Boltzmann : une méthode déterministe par intégration numérique. Comme discuté, l'équation de Boltzmann n'a pas de solution analytique pour des cas non triviaux. Une intégration numérique par maillage du domaine est donc utilisée. La méthode des éléments finis, par exemple, peut être utilisée pour réaliser la discrétisation du problème et la description de chacun des éléments discrets. Dans sa forme complète, cette méthode prend rigoureusement en compte les hétérogénéités, en admettant que la grille soit assez fine, ainsi que les différences quant au comportement des particules dans les régions de différentes compositions et donc aux variations de sections efficaces décrivant ces phénomènes. Les approximations déjà discutées pour l'algorithme de faisceau pinceau peuvent aussi être utilisées pour faciliter l'intégration. L'équation de Boltzmann sera finalement décrite comme un système d'équations linéaires à résoudre. Les principales failles de cette méthode sont la durée des calculs qui est prohibitive (plusieurs jours dans certain cas) ainsi que la quantité de mémoire requise pour effectuer la résolution du système d'équations décrivant le problème. Ce type de calcul donne aussi lieu à des erreurs systématiques de discrétisation et d'inversion de matrice.

1.2 Calcul de dose par simulation Monte Carlo

La méthode de Monte Carlo a vu le jour au laboratoire de Los Alamos suite aux travaux de Ulam, Metropolis et Von Neumann. La méthode de Monte Carlo a pour but de résoudre un système (ou une intégrale) par la répétition d'une expérience de façon stochastique jusqu'à ce que le résultat converge (Bielajew, 2001). Une autre façon de la présenter est de dire que l'on peut résoudre un système macroscopique à l'aide d'une série d'expériences sur les interactions microscopiques qui forment le système. Finalement, la méthode de Monte

Carlo se veut un outil pour trouver le comportement moyen d'un système à travers plusieurs événements indépendants et aléatoires. Dans le contexte de la présente étude, la méthode de Monte Carlo établit le parcours moyen des particules physiques pour en déduire la répartition d'énergie moyenne dans un milieu. Les particules sont guidées par des quantités physiques connues : les probabilités d'interactions, exprimées sous forme de distributions de probabilité. La combinaison de ces distributions et d'un générateur de nombres pseudo-aléatoires est à la base de la génération de trajets aléatoires pour les particules.

L'approche par Monte Carlo ne dépend pas de la nature stochastique du problème. En effet, des problèmes déterministes peuvent aussi être résolus par simulation de Monte Carlo. Le seul requis est que le problème puisse s'exprimer comme étant une fonction des nombres aléatoires utilisés par la méthode de Monte Carlo (James, 1980). La méthode de Monte Carlo se retrouve dans plusieurs sphères des sciences physiques : astronomie, chimie quantique, météorologie, microbiologie, etc.

1.2.1 Notions de probabilité nécessaires pour modélisation Monte Carlo

Quelques notions de probabilité sont nécessaires avant de détailler la méthode de Monte Carlo et comment elle peut s'appliquer au problème du transport d'énergie. Une revue plus en détail peut être trouvée dans (Feller, 1968).

La densité de probabilité (pdf) est utile pour définir la probabilité qu'une variable aléatoire continue, X, soit à l'intérieur d'un intervalle dx :

$$P(X|x_1 < X < x_1 + dx) = p(x_1)dx \tag{1.24}$$

Les propriétés d'une pdf, découlant directement des principes fondamentaux des probabilités, sont :

$$p(x) \geq 0, \tag{1.25}$$

$$\int_a^b p(x)dx = 1, \tag{1.26}$$

où le domaine de validité de p est $x\epsilon[a,b]$. La distribution de densité cumulative (CDF) se définit comme :

$$F(x) = \int_{-\infty}^{x} p(x')dx' \ = P(X < x) \tag{1.27}$$

$$\lim_{x\to\infty} F(x) = 1, \tag{1.28}$$

et de cette relation découle

$$p(x) = \frac{dF(x)}{dx}. \tag{1.29}$$

L'espérance et la variance sont deux quantités utiles lors de l'étude d'un problème par simulation de Monte Carlo. L'espérance d'une variable aléatoire X se définit comme

$$E[X] = \int_{\Omega} xp(x)dx, \tag{1.30}$$

et sa variance comme

$$\begin{aligned} var[X] &= E\left[(X - E[X])^2\right] \\ &= E[X^2] - E[X]^2. \end{aligned} \tag{1.31}$$

L'écart-type, qui sera aussi appelé incertitude, se définit finalement comme

$$\sigma(X) = \sqrt{var[X]}. \tag{1.32}$$

1.2.2 Propriétés et quantités d'intérêt

Au sens formel, la méthode de Monte Carlo peut s'exprimer comme une intégration (James, 1980). L'intégrale

$$I = \int_{\Omega} f(x)dx, \tag{1.33}$$

où f est une fonction quelconque, peut s'évaluer à l'aide de l'intégration de Monte Carlo suivante en générant un ensemble d'échantillons aléatoires à partir de la pdf du problème :

$$\overline{f} = \frac{1}{N} \sum_{i}^{N} \frac{f(X_i)}{p(X_i)}, \tag{1.34}$$

où N est le nombre d'échantillons (historique) générés. De façon moins générale mais plus près du problème qui nous occupe dans cette thèse, la simulation de Monte Carlo nous aidera à régler le problème suivant : nous cherchons une quantité d'intérêt, inconnue, μ, à partir d'un ensemble d'échantillons aléatoires $\{X_i\}$. La moyenne échantillonnée, $\overline{X}$, se définit comme :

$$\overline{X} = \frac{1}{N} \sum_{i=0}^{N} X_i. \tag{1.35}$$

Cette dernière quantité est appelée *estimateur* de la moyenne réelle, μ.

Deux propriétés importantes d'un estimateur sont le biais et la consistance. Le biais, B, est défini comme étant l'espérance de l'erreur, Er_{MC}, de la simulation de Monte Carlo :

$$Er_{MC} = \overline{X} - \mu, \tag{1.36}$$

$$B[\overline{X}] = E[Er_{MC}]. \tag{1.37}$$

L'estimateur est non-biaisé si le biais tend vers 0 pour N non nul. La consistance de l'estimateur indique que son Er_{MC} tend vers 0. Une preuve que l'estimateur $\overline{X}$ est non biaisé est décrite en Annexe II. Les conditions suivantes sont suffisantes pour prouver un estimateur consistant : 1) il est non biaisé et 2) sa variance tend vers 0 avec un nombre d'échantillons N croissant.

La variance permet, ultimement, de quantifier la confiance que l'on peut avoir envers les résultats obtenus. La définition de variance peut être appliquée à l'équation 1.34 pour trouver la variance associée en fonction du nombre d'historiques simulés (Veach, 1997)

$$\begin{aligned} var[\overline{X}] &= var\left[\frac{1}{N}\sum_{i=1}^{N} X_i\right] \\ &= \frac{1}{N^2}\left[\sum_{i=1}^{N} var(X_i)\right] \\ &= \frac{\sigma^2}{N} \end{aligned} \tag{1.38}$$

où la formule de Bienaymé a été utilisée puisque nous sommes en présence d'une somme de variables non corrélées :

$$var\left(\sum_{i=1}^{n} X_i\right) = \sum_{i=1}^{n} var(X_i). \tag{1.39}$$

L'estimateur de moyenne $\overline{X}$ est donc consistant. L'équation 1.38 démontre que pour diminuer l'incertitude statistique $\sqrt{var[\overline{X}]}$ de N, il faut simuler N^2 fois plus d'historiques. Ce résultat a de l'importance puisqu'il vient en quelque sorte borner l'incertitude qu'il est possible d'obtenir de façon pratique.

Puisque l'écart type statistique, σ, nous est aussi inconnu, nous devons utiliser un estimateur de variance échantillonnée s^2. Un estimateur non biaisé de s^2 est :

$$s^2 = \frac{1}{N-1}\sum_{i=1}^{N}(X_i - \overline{X})^2. \tag{1.40}$$

Une preuve démontrant que cet estimateur est non biaisé est donnée en Annexe II.

1.2.2.1 Convergence

Le résultat de l'Eq. 1.38 montre que le taux de convergence de la méthode est indépendant du nombre de dimensions et est proportionnel à $1/\sqrt{N}$. Il s'agit là de l'avantage principal de la méthode de Monte Carlo : de pouvoir intégrer des domaines de dimension élevée avec la même aisance que les problèmes avec domaine de petite dimension. En effet, les problèmes de

grande dimension n'ont parfois pas de solution analytique, comme c'est le cas pour l'équation de Boltzmann, où il devient peu pratique de la résoudre avec une quadrature numérique.

Une intégration numérique déterministe se fait habituellement par quadrature numérique. Dans sa forme la plus simple, l'intégration se résume à :

$$I = \int_a^b f(x)dx = \sum_{i=0}^{N} \int_{x_{i-}}^{x_{i+}} f(x_i)dx, \tag{1.41}$$

où

$$\begin{aligned} x_{i-} &= a + \frac{i(b-a)}{N} - \frac{b-a}{2N} \\ x_{i+} &= a + \frac{i(b-a)}{N} + \frac{b-a}{2N}. \end{aligned} \tag{1.42}$$

Chacune des sous intégrales peut se résoudre par une méthode d'intégration numérique, notamment la méthode du trapèze :

$$\int_{x_{i-}}^{x_{i+}} f(x_i) = (x_{i+} - x_{i-}) \frac{(f(x_{i+}) - f(x_{i-}))}{2}. \tag{1.43}$$

En remplaçant dans Eq. 1.41, l'intégration devient :

$$\hat{I} = \frac{a-b}{N}(f(x_0) + 2f(x_1) + ... + 2f(x_{x-1}) + f(x_n)) + E \tag{1.44}$$

où E, le terme d'erreur, est défini par (Fortin, 1996)

$$E = \left| I - \hat{I} \right| \leq C \frac{||f''||}{N^2} \tag{1.45}$$

où C est une constante. Cette construction en une dimension peut se généraliser en D dimensions pour donner un terme d'erreur (James, 1980) :

$$E \leq C \frac{||f''||}{N^{2/D}} \tag{1.46}$$

En rappelant que la convergence de la méthode de Monte Carlo est en $1/\sqrt{N}$: il est donc

évident qu'il existe toujours une complexité, en terme de nombre de dimensions, où la méthode de Monte Carlo devient avantageuse.

Un autre théorème important pour qualifier le résultat d'une simulation de Monte Carlo est le théorème de limite centrale (James, 1980). Le théorème indique que la somme d'un grand nombre de variables aléatoires est distribuée selon une distribution normale. Cette distribution est complètement définie à l'aide de la moyenne $\overline{X}$ et son incertitude σ_X. À partir de cette propriété, qui s'applique lorsque N est suffisamment grand, il est possible de poser que la véritable valeur μ se situe à l'intérieur d une distance $\pm\sigma_X$ de $\overline{X}$ avec une probabilité de 68.3%, à l'intérieur d'une distance de $\pm 2\sigma_X$ de $\overline{X}$ avec une probabilité de 95.4% et à l'intérieur d'une distance de $\pm 3\sigma_X$ de $\overline{X}$ avec une probabilité de 99.7%.

1.2.2.2 Échantillonage

La méthode pour générer des historiques consiste à interroger les distributions de probabilité qui décrivent le problème. Ce concept se nomme l'échantillonnage. La méthode choisie pour échantillonner ces distributions de probabilité dépend de la nature de la distribution.

La méthode d'échantillonnage directe (Bielajew, 2001) se base sur le fait que l'inverse d'une CDF peut être facilement trouvée de façon analytique. Pour utiliser la méthode, nous avons besoin d'un nombre aléatoire, ξ, distribué de façon uniforme dans l'intervalle [0,1], c'est-à-dire :

$$F(\xi) = \xi, \qquad si \;\; 0 \leq \xi \leq 1 \tag{1.47}$$

À l'aide de cette variable aléatoire, la CDF peut être échantillonnée puisque celle-ci a aussi la caractéristique d'être définie sur l'intervalle [0,1]. Mathématiquement : $\xi = F(x)$ ce qui revient à trouver la valeur de x pour laquelle cette équation est vérifiée. Si la CDF est inversible, nous pouvons trouver $x = F^{-1}(\xi)$ et x a donc été échantillonné aléatoirement.

Lorsque l'inverse de la CDF ne peut pas être trouvé facilement, une deuxième méthode d'échantillonnage peut être utilisée : la méthode par rejet (Salvat et al., 2006). La méthode

demande de pouvoir échantillonner rapidement (par la méthode directe par exemple), une distribution $g(x)$ tel que $Cg(x) > f(x)$ en tout point, où C est une constante. Les étapes de la méthode sont les suivantes :

1. Échantillonner une variable aléatoire x de $Cg(x)$
2. Générer une variable aléatoire $\xi \epsilon U[0,1]$
3. Si $\xi \leq \frac{f(x)}{Cg(x)}$ accepter x, sinon revenir au point 1.

Il est donc facile de voir que l'efficacité de cette méthode n'est pas toujours complète et de par le principe d'enveloppe l'échantillon sera accepté avec un ratio égal à l'intersection de $f(x)$ et $Cg(x)$.

1.2.2.3 Efficience et réduction de variance

Une quantité d'intérêt pour évaluer la qualité d'une approche Monte Carlo est la notion d'efficience :

$$\epsilon = \frac{1}{T\sigma^2}, \tag{1.48}$$

où T est le temps requis pour calculer $\overline{X}$. La conception d'une simulation de Monte Carlo vise à créer des estimateurs efficients. L'ensemble des techniques pour augmenter l'efficience d'un estimateur se nomme réduction de variance. L'ajout de ces techniques rend parfois la simulation dépendante du problème, ce qui enlève de la généralité à la méthode. Les techniques de réductions de variance ne feront pas l'objet de la présente étude. Une revue de quelques-unes de ces techniques est faite par Kawrakow (Kawrakow and Fippel, 2000).

1.2.3 Génération de parcours en transport d'énergie par simulation de Monte Carlo.

La méthode de Monte Carlo pour le transport d'énergie se distingue d'une méthode déterministe par le fait que l'équation de Boltzmann n'est jamais réellement solutionnée. En effet, il est possible, comme il sera présenté ici, d'exprimer le problème de transport d'énergie par la

méthode de Monte Carlo sans se référer à l'équation de Boltzmann.

Ce que l'approche par simulation de Monte Carlo accomplit est de faire le comptage d'une certaine quantité d'intérêt. Cette quantité peut être, par exemple, le nombre d'interactions, l'énergie déposée, la distance parcourue, etc..

1.2.3.1 Généralités

Le problème de transport d'énergie dans un milieu se prête bien à la deuxième illustration de la résolution d'un problème par simulation de Monte Carlo introduite en début de chapitre : la répartition de la dose (de l'énergie par unité de masse) dans chaque région, notre processus macroscopique $D(\vec{x})$, peut se trouver en simulant les éléments microscopiques du problème soient les parcours des particules à l'intérieur du milieu. La simulation de Monte Carlo revient donc à lancer un certain nombre de particules primaires, N, et de voir comment ces N particules se répartissent dans le milieu et où elles déposent leur énergie. L'échantillonnage du parcours ramène à un processus de Markov (Salvat et al., 2006) où le parcours total de la particule est divisé en un ensemble d'interactions, de déviations et de dépôts d'énergie indépendant les uns des autres. L'échantillonage du prochain sous-parcours dépend donc seulement des propriétés actuelles de la particule et non des interactions qu'elle a subies auparavant.

L'échantillonnage des parcours de particule se fait par échantillonnage des différentes sections efficaces différentielles présentées au Chapitre 1. En effet, ces dernières permettent de définir l'orientation de la particule après interaction, l'énergie transférée à une particule secondaire ainsi que l'orientation de cette particule secondaire, ce qui constitue l'ensemble des informations dont nous avons besoin pour échantillonner un parcours complet.

Les particules à simuler évoluent dans un environnement à 6 dimensions : trois de position $\vec{x} = \{x, y, z\}$, deux d'orientation $\vec{\Omega} = \{\theta, \phi\}$ et une d'énergie E. Chacune des interactions altère l'une ou l'autre ou toutes ces composantes.

La simulation de Monte Carlo décrit le parcours chacune des particules primaires ainsi que toutes les particules secondaires créées au cours des interactions jusqu'à ce que chacune d'entre elles sortent de la région étudiée ou que leur énergie soit sous une certaine limite dictée par l'utilisateur. Cette dernière condition a un aspect pratique mais elle est aussi une nécessité puisque les sections efficaces divergent, c'est-à-dire la distance entre deux interactions successives tend vers 0, lorsque l'énergie tend vers 0. Cette approximation implique nécessairement une certaine perte d'exactitude puisqu'une particule qui aurait déposé une partie de son énergie en $\vec{x}$ a plutôt été absorbée en $\vec{x'}$. Pour illustrer cette approximation, avec une limite en énergie pour les électron $E_{th} = 10$ keV un électron parcourra en moyenne 10^{-4} cm dans l'eau avant d'être complètement absorbé (Berger et al., 2006). Cette quantité se calcule à l'aide de la notion de portée :

$$R = \int_0^E \left[\frac{S(E)}{\rho(\vec{x})}\right]^{-1} dE. \tag{1.49}$$

Pour les photons, le résultat est quelque peu différent puisque ceux-ci ne déposent pas directement d'énergie mais génèrent des électrons secondaires qui eux vont déposer de l'énergie. Avec une limite d'énergie pour les photons de $P_{th} = 1$ keV la distance moyenne à la prochaine interaction (MFP) est de 2.5^{-4} cm et l'énergie maximale de l'électron qui pourrait être créé à cet endroit est de 1 keV.

Ces approximations ont donc un impact non nul sur l'exactitude du calcul Monte Carlo mais leur impact dépend de l'exactitude requise par l'application. Par exemple, pour le calcul sur une grille voxélisée où chaque voxel a une taille de 0.5 cm, un électron de 10 keV a peu de chance de sortir de son voxel et donc l'approximation est sans impact. Par contre, dans une géométrie paramétrée où certaines composantes peuvent avoir une taille sous-millimétrique, il est important de garder E_{th} et P_{th} aussi bas que possible.

1.2.3.2 Source de particules

La simulation démarre par l'émission de particules à partir d'une source, le terme $S(\vec{x}, \vec{\Omega}, E)$ de l'Eq 1.19. Ce terme peut être situé à l'intérieur ou à l'extérieur du domaine où le dépôt de dose a lieu.

Pour un problème de radiothérapie externe, où la source de particules réelle est un accélérateur linéaire, la source virtuelle est placée à l'extérieur du domaine. Le degré de fidélité de la représentation de l'accélérateur linéaire est variable. À un extrême, une source ponctuelle et monoénergétique peut être utilisée pour remplacer l'accélérateur linéaire. À l'autre extrême, une représentation complète de la géométrie de la tête de l'accélérateur linéaire et une source réaliste d'électrons frappant une cible en tungstène pour permettre la génération de photons de *bremsstrahlung* peut être utilisée. Cette dernière possibilité représente à elle seule une simulation de Monte Carlo de transport d'énergie. Le résultat de cette simulation peut être utilisé directement par une autre simulation faisant le transport à l'intérieur du patient ou stocké hors ligne dans un fichiers d'*espace de phase*, stockant l'énergie, la position ainsi que l'orientation de toutes les particules croisant un plan donné. Dans tous les cas, les particules générées doivent être projetées sur la frontière de la région de simulation.

Pour un problème de curiethérapie, où la source de particules réelle est une source radioactive placée à l'intérieur du patient, la source virtuelle est à l'intérieur du domaine de simulation. La source peut encore une fois être simple : une source isotrope et monoénergétique ; ou complexe : une représentation complète de la géométrie de la source et du spectre du radionucléide.

1.2.3.3 Distance à la prochaine interaction

Comme mentionné précédemment, le parcours macroscopique d'une particule se définit comme un ensemble de sous-parcours microscopiques indépendants les uns des autres, tel un processus de Markov. Il est donc possible de définir le parcours d'une particule à partir d'un état, un triplet $\{\vec{x}, \vec{\Omega}, E\}$. La discussion qui suit s'applique de façon absolue tant aux particules

chargées qu'aux particules non chargées mais il sera expliqué au Chapitre 3 que le parcours des particules chargées est trouvé en pratique d'une façon différente. À partir de l'état, la distance à la prochaine interaction de la particule est trouvée en fonction du coefficient d'atténuation total, μ, détaillé en section 1.1.1. De μ, le MFP est défini :

$$\lambda = \mu^{-1}, \tag{1.50}$$

et correspond à la distance espérée entre deux interactions étant donné un milieu homogène de coefficient d'atténuation $\mu(\vec{x}) = \mu$ et une particule avec un état $\{\vec{x}, \vec{\Omega}, E\}$. Dans une approche Monte Carlo, la distance à la prochaine interaction est échantillonnée à partir de la pdf de la distance du libre parcours. Cette distribution se trouve de la façon suivante. Admettons $dP(s)$ la probabilité d'interagir dans un élément de distance infinitésimal. Cette quantité est, par définition, égale à

$$dP(s) = \mu ds. \tag{1.51}$$

Admettons aussi $P_0(s)$ la probabilité qu'une particule ait parcouru $s_0 \rightarrow s$ sans interagir, la probabilité de non-interaction. La variation de probabilité de non-interaction si la particule parcours un ds supplémentaire est :

$$dP_0(s) = -P_0(s)\mu ds. \tag{1.52}$$

En regroupant les termes semblables, nous obtenons :

$$\frac{dP_0(s)}{P_0(s)} = -\mu ds, \tag{1.53}$$

et après intégration :

$$P_0(s) = e^{-s\mu}. \tag{1.54}$$

Avec ces deux définitions, il est possible de poser la quantité suivante : la probabilité que la particule ait parcouru une distance $s_0 \rightarrow s$ sans interagir et qu'elle interagisse dans le prochain ds :

$$P_0(s)\mu ds = \mu e^{-s\mu} ds. \tag{1.55}$$

La pdf de la distance de libre parcours est donc :

$$f(s) = \mu e^{-s\mu}. \tag{1.56}$$

L'Eq. 1.56 peut être échantillonnée pour trouver la distance à la prochaine interaction pour chacun des sous-parcours avec la méthode directe d'échantillonnage. La CDF de Eq. 1.56 est $F(s) = 1 - e^{-\mu s}$ et suivant la méthode directe :

$$\xi = F(s), \tag{1.57}$$

où ξ est une variable aléatoire répartie uniformément dans l'intervalle (0,1], pour finalement échantillonner s :

$$s = \lambda \, log(\xi). \tag{1.58}$$

Par la suite, la particule peut être avancée à la position de la prochaine interaction suivant

$$\vec{x'} = \vec{x} + s\Omega, \tag{1.59}$$

et son énergie modifiée selon l'échantillonage de l'interaction. À cette étape, il est possible que

la position $\vec{x'}$ soit hors du domaine étudié. Le parcours de cette particule est alors terminé.

1.2.3.4 Transport en milieu hétérogène

Il faut rappeler que l'Eq. 1.56 prend en hypothèse un milieu homogène et infini. Or, le transport de particules peut se faire dans des milieux hétérogènes où différents matériaux de propriétés d'atténuation différentes sont présents. L'hypothèse d'un facteur d'atténuation constant pour tout le milieu ne tient alors plus. Admettant la situation suivante : la distance à la prochaine interaction, s, est telle que la particule parcourrait une distance s_1 dans la région r_1 de facteur d'atténuation μ_1 et une distance s_2 dans la région r_2 de facteur d'atténuation μ_2. La distance à la prochaine interaction doit donc être corrigée pour tenir compte de ce changement de milieu. Une possibilité de correction est la suivante (Leppanen, 2007) :

$$e^{-s_2\mu_2} = e^{-(s-d)\mu_1} \Longleftrightarrow \mu_2 s_2 = (s-d)\mu_1 \tag{1.60}$$

où d est la distance de la position courante à la limite de la région r_1. À partir de cette relation, la distance corrigée entre la position actuelle et la position de prochaine interaction, en tenant compte des milieux différents, se calcule comme :

$$s = d + (s_1 - d)\frac{\mu_1}{\mu_2}. \tag{1.61}$$

Cette stratégie requiert par contre l'emploi d'un module de géométrie capable de trouver la distance entre la position actuelle $\vec{x}$ et la position de la prochaine intersection étant donnée l'orientation actuelle $\vec{\Omega}$.

Une stratégie différente a été développée par Woodcock (Woodcock et al., 1965) et consiste à rendre le milieu homogène, par rapport à l'atténuation, de façon artificielle. Pour ce faire, le coefficient d'atténuation est majoré à l'aide d'un coefficient d'atténuation μ_f, lié à un type d'interaction fictif. La valeur de coefficient d'atténuation est variable pour chaque région et est décidée de façon à ce que l'équation suivante soit respectée pour une valeur μ_0 unique à

l'ensemble du domaine :

$$\mu_0 = \mu(\vec{x}, E) + \mu_f(E). \tag{1.62}$$

L'Eq. 1.58 est alors utilisée avec $\lambda = \mu_0^{-1}$. L'avantage de cette technique est que le problème de traitement des régions hétérogènes est éliminé. Le désavantage est que la méthode devient inefficace si un milieu est très fortement atténuant par rapport au reste de la géométrie. En effet, plus la différence entre la valeur au point le plus atténuant du milieu, caractérisé par $\mu(\vec{x}, E) = \mu_0$, et la valeur moyenne du coefficient d'atténuation du domaine, plus le nombre d'interactions fictives sera élevé. Un nombre élevé d'interactions fictives a pour conséquence qu'un plus grand nombre de pas que nécessaire a été simulé pour terminer le parcours d'une particule, ce qui diminue l'efficience de la simulation.

1.2.3.5 Choix de l'interaction

À la position échantillonnée précédemment, le choix de l'interaction se fait par échantillonnage des données de sections efficaces des différentes interactions possibles pour le type de particule sur le point d'interagir. Pour ce faire, il est nécessaire de connaître la section efficace totale pour cette énergie et cette position :

$$\sigma_t(\vec{x}, E) = \sum_{i=1}^{n} \sigma_i(\vec{x}, E), \tag{1.63}$$

où n est le nombre d'interactions possibles pour ce type de particule. Par la suite, les facteurs de branchement pour chaque type d'interaction i sont calculés :

$$F(i) = \frac{\sum_{j=1}^{i} \sigma_j(\vec{x}, E)}{\sigma_t(\vec{x}, E)}. \tag{1.64}$$

Finalement, l'échantillonnage du type d'interaction se trouve à l'aide d'un nombre $\xi \epsilon U(0, 1)$ tel que :

$$F(i-1) < \xi < F(i). \tag{1.65}$$

1.2.3.6 Simulation de l'interaction

La simulation des diverses interactions sera définie plus en détail au Chapitre 3. Ultimement, aucune avancée n'est proposée dans ce travail quant à la façon d'échantillonner une interaction et le développement mathématique du traitement de toutes les interactions peut être trouvé dans les références aux manuels des plateformes Monte Carlo généralistes.

Outre le développement physique menant à la simulation de chaque interaction, chaque interaction peut mener à un nombre d'événements parmi quatre possibilités : la modification de l'énergie de la particule incidente, la modification de l'orientation de la particule incidente, l'annihilation de la particule incidente et la création d'une particule secondaire.

1.2.3.7 Collecte de résultats

La collecte de résultats s'effectue sur une grille discrète positionnée sur une région d'intérêt. Dans une simulation de type analogue, la donnée enregistrée dans chaque cellule de la grille est l'énergie déposée. Comme mentionné au paragraphe précédent, chaque interaction a la possibilité de déposer une portion de l'énergie de la particule dans le milieu environnent. C'est cette quantité que la simulation étudie.

Pour chaque cellule de la grille se trouve un accumulateur de la forme :

$$I = \sum_{i=1}^{N} I_i, \tag{1.66}$$

où les I_i correspondent, dans ce cas-ci, à l'énergie déposée par chaque événement. À partir de l'Equation 1.34, on se rappelle que la quantité d'intérêt est la moyenne de l'accumulateur :

$$\bar{I} = \frac{1}{N} \sum_{i=1}^{N} I_i. \tag{1.67}$$

La précision statistique de la simulation, en fonction du nombre d'événements simulé, se trouve à l'aide de l'Equation 1.32.

1.3 Méthodes existantes de simulation Monte Carlo rapides pour calcul de dose

Les programmes de simulation de Monte Carlo pour le transport d'énergie par particules chargées et non chargées sont majoritairement de grands projets ayant évolués sur quelques décennies. Les plus utilisés sont GEANT4 (Agostinelli et al., 2008), PENELOPE (Salvat et al., 2003) et EGSnrc (et ses ancêtres) (Kawrakow and Rogers, 2000). Une revue des possibilités de chacun serait beaucoup trop longue pour le présent document et nous nous intéressons principalement à des techniques rapides de simulation. De ces programmes généraux, quelques sous-projets ont été développés dans le but d'accélérer particulièrement les calculs de dose tout en gardant un niveau d'exactitude acceptable. De plus, des techniques de réduction de variance ont été développées pour réduire davantage le temps de calcul. Ces logiciels rapides sont présentés dans les sous-sections 1.3.1 et 1.3.2.

1.3.1 DPM

DPM a été développé par Sempau *et al.* (Sempau et al., 2000). Sa plage de validité est, comme mentionné, restreinte aux énergies utilisées lors des traitements de radiothérapie, soit [100keV-20MeV]. DPM est une adaptation du programme PENELOPE.

Au point de vue des interactions des photons, DPM ne considère pleinement que les interactions de type Compton puisque ce sont les plus fréquentes dans la plage d'énergie utilisée. L'effet photoélectrique est approximé en assumant que toute l'énergie est déposée localement, au site d'interaction. C'est donc dire qu'il n'y a pas de génération d'électron secondaire ou de radiation de relaxation de l'atome. La création de paires est aussi fortement approximée puisqu'elle ne survient que dans la toute fin de la plage énergétique et principalement pour des matériaux à haut numéro atomique, éléments peu fréquents en calcul de dose. L'approximation faite est que les deux particules générées ont une énergie aléatoire (mais que leur total

est équivalent à l'énergie du photon absorbé) et qu'elles voyagent dans la même direction que le photon incident. Les deux particules sont aussi simulées comme des électrons jusqu'à ce qu'une interaction survienne, dans quel cas une des deux est aléatoirement choisie pour se comporter comme un positron.

L'algorithme de traçage de Woodcock (Woodcock et al., 1965) est utilisé pour l'aspect géométrique du problème de transport de photons.

Pour les électrons, DPM utilise un modèle d'historique condensé (Berger, 1963) (CH, *Condensed History*) de type 2, ou mixte. Cette facette du transport de particules sera explicitée au Chapitre 3.

L'article portant sur DPM fait état de trois techniques peu répandues (à ce moment) :

1. L'utilisation du formalisme de diffusions multiples de Bielajew et Kawrakow (Kawrakow and Bielajew, 1998). Ce formalisme permet de garantir que la simulation converge vers la solution pour une longueur de saut d'électron donné ;
2. L'utilisation de grandes valeurs pour la distance parcourue dans une étape du transport de l'électron, pouvant traverser plusieurs voxels.
3. L'utilisation de la méthode *random hinge* présente dans PENELOPE qui divise un saut d'électron en deux parties pour modéliser le déplacement latéral et la déviation angulaire subis lors du saut. La méthode est modifiée pour accommoder de grandes valeurs à l'intérieur d'un saut d'électron.

La mécanique de transport d'électrons de DPM est aussi novatrice avec l'utilisation d'un système de carburant consommable par les particules. Pour chaque type d'interaction d'électron (Møller, *bremsstrahlung*, diffusion multiple), un carburant initial est choisi aléatoirement suivant les facteurs d'atténuation de chaque type d'interaction. Par exemple :

$$t_i = -\mu_i ln\xi \tag{1.68}$$

À chaque traversée de voxel, le carburant est réduit en fonction du voxel traversé. Lorsque le

carburant d'une interaction est complètement utilisé, une interaction dudit type est simulée et le carburant à nouveau choisi aléatoirement. Cette technique requiert donc de transporter les électrons voxel par voxel, mais sans traitement explicite du changement de valeur de la longueur du saut d'électron à chaque voxel. En effet, la valeur de t_i est calculée une seule fois, lorsque le plein de carburant est fait. Le saut se fait donc en considérant le voxel en cours, mais sans recalcul de t_i même si celle-ci dépend de μ qui peut varier de voxel en voxel.

1.3.2 VMC et XVMC

VMC (Kawrakow et al., 1996) et XVMC (Fippel, 1999) sont deux programmes aussi développés dans le but de calculer la dose dans une application en radiothérapie, ce que suggère aussi son acronyme *Voxel based Monte Carlo*. Cette famille de programme a été développée majoritairement à partir de la physique présente dans EGSnrc.

Pour les électrons, VMC utilise plusieurs approximations pour augmenter la rapidité du calcul. Tout d'abord, les interactions de *bremsstrahlung* déposant peu de dose sont approximées par une technique rappelant le faisceau pinceau. Des données pré-calculées à l'intérieur d'un fantôme d'eau sont adaptées à la nouvelle géométrie du patient en cours. Cette technique élimine aussi la simulation des photons secondaires créés durant l'interaction puisque leur dépôt d'énergie est aussi inclus dans la source de données hors ligne.

Une technique de réutilisation de l'historique des particules est utilisée. En assumant que deux particules démarrent à des positions différentes et qu'il est peu probable qu'une particule vienne contribuer à la dose dans la région où l'autre particule évolue, seulement un historique de particule est généré. Cet historique est alors translaté pour être appliqué à la deuxième particule. Cette technique permet de sauver du temps CPU en ne requérant qu'un seul échantillonnage des diverses distributions de probabilité cumulatives associées au problème. Cette technique requiert par contre de stocker les différentes étapes de l'historique des électrons. Pour une application sur GPU, une approche où une donnée est calculée une seule fois et utilisée par plusieurs processus peut s'avérer néfaste lorsque le traitement desdites

données n'est pas homogène. C'est ce que nous trouvons ici puisque des tests doivent être faits pour trouver quel historique peut s'appliquer à quelles particules. L'approche de réutilisation d'historiques peut aussi être employée pour appliquer des historiques obtenus dans un milieu homogène vers un milieu hétérogène en mettant à l'échelle par rapport à la constitution des voxels traversés.

Un mélange de méthodes empiriques et d'essais et erreurs est utilisé pour définir la longueur t maximale que peut parcourir un électron lors d'un saut d'électron. Les équations (13-16) de l'article de Kawrakow (Kawrakow et al., 1996) présentent cette technique.

Tel que déjà mentionné, plusieurs propriétés du matériau (Z, pouvoir d'arrêt S^0, densité ρ) doivent être connues pour effectuer la simulation de Monte Carlo. Or, la seule source de données d'entrée est le volume de voxels contenant des données CT. Il est donc nécessaire d'extraire, pour chaque voxel, le matériau et la densité du voxel à l'aide de son nombre Hounsfield acquis par imagerie médicale. Leur technique ne décompose pas directement en différents matériaux mais permet plutôt d'obtenir la valeur d'intérêt dans ce cas (la densité électronique) à partir d'une courbe ajustée.

Les photons ne sont pas incorporés à VMC. Rappelons que la production de photons secondaires dus au *bremsstrahlung* est ignorée.

Du point de vue de la rapidité d'exécution, VMC est environ 35 fois plus rapide que EGS4.

XVMC présente un nouveau mode de génération d'électrons secondaires dus au faisceau primaire de photons en utilisant une technique similaire à celle utilisée dans un calcul de convolution/superposition (Mackie et al., 1985). Une carte de TERMA initiale est générée par tracé de rayons à travers le volume de voxel et des électrons sont mis en mouvement à chaque voxel selon la valeur de TERMA. Le désavantage de cette technique est que la simulation est limitée à l'utilisation de sources simples et qu'il est impossible de démarrer la simulation à l'aide d'informations fournies dans un espace de phase préalablement calculé. Cet espace de phase contient la description de plusieurs millions de particules. Ces particules ont, par exemple, été produites lors de la simulation d'un accélérateur linéaire et correspondent

aux particules qui sont sur le point d'interagir avec le volume de voxels.

Tout comme dans DPM, l'effet photoélectrique a d'abord été ignoré (Fippel, 1999) pour ensuite être rajouté à XVMC (Kawrakow and Rogers, 2000). Une justification du choix d'ignorer l'effet photoélectrique est présentée par (Kawrakow and Fippel, 2000) où il est indiqué que la différence est en deçà d'un point de pourcentage pour les énergies de 1-2 MeV et complètement négligeable pour les plus hautes énergies. Du point de vue de la production de paire, le positron généré est transporté comme un électron.

Dans l'article subséquent de Kawrakow et Fippel, des techniques de réduction de variance sont présentées. Ces techniques ont pour but d'améliorer l'efficience de la simulation de Monte Carlo.

La première technique de réduction de variance est *l'interaction forcée*. Dans le transport de photons, plusieurs photons vont traverser le fantôme sans même interagir. L'interaction obligatoire veut identifier les photons qui vont interagir et ne transporter que ceux-ci. La conclusion trouvée est que cette technique est avantageuse si l'énergie incidente des photons est supérieure à 6 MeV pour un fantôme de 25 cm. Cette technique semble par contre peu avantageuse pour une implémentation SIMD (instruction unique, données multiples (*Single Instruction Muliple Data*) puisque tous les photons d'un vecteur SIMD devraient être soit actifs soit mis de côté pour qu'un gain soit possible.

La deuxième technique de réduction de variance a déjà été discutée. Il s'agit de l'étape primaire de TERMA.

La troisième méthode consiste à utiliser un générateur de nombres quasi aléatoires de type Sobol plutôt qu'un générateur de nombres pseudo-aléatoires. Ces générateurs de nombres ont la propriété de mieux remplir un espace de d dimension uniformément qu'un générateur de nombres pseudo-aléatoires. Des gains de 1.1 à 2 ont été trouvés lorsque ces générateurs sont utilisés pour définir la source et la longueur de saut s. Cette technique pourrait être utilisée dans un environnement SIMD GPU mais requerrait une utilisation supplémentaire des ressources pour stocker le générateur de nombres pseudo-aléatoires.

La technique de réduction de variance dite de la roulette russe vise à réduire le temps CPU passé à simuler des photons secondaires et les électrons qu'ils mettent en mouvement. Le photon sera simulé seulement si un nombre aléatoire entre zéro et un est plus petit que la valeur $\frac{1}{f_{weight}}$ où f_{weight} est un poids associé à une particule. Cette technique n'a pas apporté de gain en efficacité puisqu'elle joue aussi, défavorablement, sur l'incertitude statistique. Lorsqu'elle est jumelée à une technique de division (*splitting*) de particules, elle permet d'obtenir des gains entre 2.5 et 5.

1.4 Plateforme matérielle et environnement de programmation

Le GPU a originellement été développé pour accélérer le calcul graphique. Il était muni d'un ensemble d'éléments de calcul fixes à l'intérieur d'un pipeline graphique fixe. Avec l'avènement de la famille GeForce 3 de NVIDIA est apparu un premier niveau de programmabilité. Toujours à l'intérieur d'un contexte de calcul graphique, un programmeur avait accès à deux des étapes du pipeline fixe : le nuanceur de sommets et le nuanceur de fragments. En 2003, soit deux ans après la venue de la GeForce 3, NVIDIA annonçait le langage Cg dans le but de simplifier, toujours dans un contexte graphique, la programmation des nuanceurs.

Il a fallu attendre la GeForce 8 de NVIDIA pour voir l'arrivée de l'architecture de nuanceur unifié (*unified shader model*). Cette architecture éliminait la distinction matérielle entre nuanceur de sommets et de fragment et mettait à la disposition des programmeurs des processeurs à usage général (mais pas aussi général que le CPU) pour y effectuer leurs calculs. Environ un an après, NVIDIA annonce le langage CUDA (Nvidia, 2010), dont le but est de permettre une utilisation du GPU pour faire du calcul général, sans la contrainte de travailler dans un contexte graphique.

Le chapitre courant ne porte pas sur la programmation des cartes graphique en général mais bien sur la programmation, en utilisant CUDA (*compute unified device architecture*), des cartes graphiques de la compagnie NVIDIA. La raison de ce choix est qu'au moment

du début de ce projet, la plateforme ouverte OpenCL [1] en était à ses débuts et n'était pas considérée assez stable pour entreprendre le présent travail.

Le reste de ce chapitre se concentre sur la plateforme matérielle, le GPU, et son interface de programmation à travers CUDA.

1.4.1 Le processeur graphique

Cette section détaillera le processeur graphique de la compagnie NVIDIA. Le premier GPU fonctionnant avec l'interface CUDA est le G80. Depuis sa sortie en 2007, trois autres modèles ont vu le jour : le G92 une évolution du G80, le GT200 qui a marqué l'avènement du calcul double précision et le GF100 qui a marqué une réorientation très visible du GPU vers un engin de calcul scientifique avec l'ajout d'éléments qui ne profitent pas au rendu graphique. Cette section portera généralement sur des aspects applicables à toutes les familles et avec certains détails applicables seulement à la famille GF100. La famille GT200 sera en grande partie ignorée.

Le design du GPU a pour but de permettre l'exécution rapide d'une charge de travail avec un très haut niveau de parallélisme de données. En effet, le travail « traditionnel » du GPU est le rendu graphique où un travail similaire doit être effectué sur un grand nombre de sommets ou de fragments. À travers les années, la distinction entre nuanceur de sommets et nuanceur de fragments est disparue pour faire place à un nuanceur unifié capable de faire des calculs généraux. C'est cette architecture unifiée qu'a amené le G80.

Le GPU peut être associé à un ensemble de processeur SIMD travaillant à résoudre un problème. En terminologie NVIDIA, le GPU est constitué de plusieurs *multiprocesseur* (MP) qui à leur tour sont constitués de *processeurs scalaires* (SP). L'unité SIMD du GPU est le MP, ce qui implique que les SP d'un MP doivent travailler au même problème, c'est-à-dire exécuter la même instruction. Par contre, chaque SP aura son propre jeu de données, ce que constitue l'approche de parallélisme de données.

1. http ://www.khronos.org/opencl/

Le GPU possède sa propre hiérarchie de mémoire. Le reste du système *hôte* communique avec le GPU à travers sa mémoire *globale*. Ce niveau de mémoire constitue la principale réserve d'espace mémoire sur la carte et est analogue à la mémoire vive du système hôte. Cette mémoire est accessible en lecture/écriture à partir de l'hôte ainsi que des SPs. La mémoire globale, dans toutes les architectures pré-GF100, n'était pas cachée, ce qui implique que la localité de données n'avait pas d'impact positif sur la bande passante. À partir du GF100, un étage de cache L2 de 768 kb a été ajouté entre la mémoire globale et le niveau des MP et un niveau L1 de 64 kb (configurable entre 48/16 kb ou 16/48 kb entre la cache automatique et la cache programmée) local à chaque MP. Cette cache permet de favoriser la réutilisation de données lorsqu'il y a localité entre les accès des différents SP ainsi que de réduire la latence associée à une lecture en mémoire globale.

C'est aussi en mémoire globale que sont stockées les textures. Les textures, avant la famille de GPU GF100/FERMI, avait la particularité d'être le seul moyen de garder une cache automatique des données accédées de la mémoire globale. Elle avait donc une grande utilité pour les accès mémoire hors d'ordre. La localité, et donc les données mises en cache, de la texture dépend de sa dimension. CUDA permet de définir des textures 1D, 2D et 3D. De plus, les textures exposent le matériel d'interpolation présent sur la carte et permettent d'avoir accès à une interpolation « gratuite » lorsque des indices non-entiers sont utilisés pour accéder à une case mémoire.

1.4.2 L'interface de programmation

Un programme voulant intégrer une composante de calcul sur GPU peut utiliser un ensemble de solutions logicielles, par exemple Brook+, AMD CTM, OpenCL et CUDA. Comme mentionné en début de chapitre, cette thèse utilise CUDA et cette section sera donc limitée à cette interface de programmation. Il est par contre intéressant de remarquer la très grande similitude entre CUDA et OpenCL en ce qui a trait au concept et au paradigme de programmation.

Une section de calcul CUDA a pour but d'exécuter un *kernel* sur une *grille*.

La *grille* représente le parallélisme du travail à être effectué sur le GPU. Cette grille se divise en un certain nombre de blocs, qui sont à leur tour composés d'un certain nombre de processus. Le processus n'est par contre pas l'unité élémentaire de parallélisme puisque le GPU est une architecture SIMD. En effet, l'unité de parallélisme élémentaire est le *warp*, constitué de 32 processus. Un bloc s'exécute sur un MP donné et, si les ressources le permettent, plus d'un bloc peuvent s'exécuter sur un MP. Par contre, un bloc ne peut se diviser sur plus d'un MP.

Le *kernel* représente une fonction devant s'exécuter directement sur le GPU. Puisque l'approche GPU est une approche SIMD, cette fonction est commune à tous les processus s'exécutant sur la grille. À l'intérieur de la fonction, il est possible d'identifier chaque processus de façon unique à l'aide des variables intrinsèques *threadIdx.*$\{x,y,z\}$ et *blockIdx.*$\{x,y\}$.

1.4.3 Applicabilité du calcul sur GPU

Le calcul sur GPU ne se prête pas à toutes les situations de calcul. La liste non exhaustive de caractéristiques suivante illustre les qualités d'une application où un passage au calcul sur GPU peut se révéler avantageux :

- large domaine de données, par exemple de grandes matrices à traiter ;
- indépendance des opérations à effectuer, par exemple chacun des calculs sur les éléments $x^s_{i=I,j=J}$ d'une matrice, pour l'itération s, ne dépend pas des autres résultats $x^s_{i\neq I,j\neq J}$;
- opérations homogènes sur les données, par exemple l'algorithme utilisé pour calculer $x^s_{i=I,j=J}$ est le même que pour calculer $x^s_{i\neq I,j\neq J}$;
- localité des calculs, par exemple le calcul de $x^s_{i=I,j=J}$ ne requiert pas l'accès à tous les éléments de x ;
- relativement peu de synchronisation nécessaire, par exemple les éléments $x^s_{i=I,j=J}$ et $x^s_{i=I+1,j=J}$ peuvent se calculer simultanément ;
- faible taux de transfert CPU-GPU, par exemple entre les itérations s et $s+1$, aucun

transfert de données n'est nécessaire.

Le non-respect de l'une de ces conditions ne garantit pas un échec de l'utilisation du GPU pour effectuer le calcul. Par exemple, la troisième condition n'est pas respectée dans la présente thèse puisque chaque particule interagira de façon stochastique. Le calcul sur GPU a été utilisé dans divers domaines, à l'intérieur et à l'extérieur de la physique médicale (par exemple (King et al., 2010; Tomov et al., 2010; Schneider et al., 2010; Wang et al., 2010; Fogal et al., 2010; Jia et al., 2010a; Jia et al., 2010b)).

1.5 Simulations sur GPU

Tel que mentionné, au cours des dernières années trois implémentations de programmes Monte Carlo ont été développés pour s'exécuter sur du matériel graphique.

1.5.1 MCML

Le premier programme Monte Carlo pour transport d'énergie fut publié par Alerstam *et al.* (Alerstam et al., 2008). Leur implémentation est basée sur un programme existant, MCML (Wang et al., 1995) et il ne fait que du transport de photons, c'est-à-dire que toutes les interactions entre les photons et le milieu qui normalement donneraient lieu à la création d'électrons secondaires déposent plutôt leur énergie localement. Cette approximation peut être acceptable en basse énergie ($\approx$<2 MeV) mais devient innaceptable en haute énergie.

En ce qui concerne leurs résultats publiés, le volume dans lequel évoluent les photons est homogène, ce qui élimine le besoin de vérifier si le photon passe d'un milieu à un autre et qui élimine aussi une source de divergence possible pour l'architecture SIMD. Avec ces conditions expérimentales, leur application est $\approx 1000\times$ plus rapide que l'équivalent CPU. Le manuel d'utilisation du programme présente des situations avec un CPU plus récent et l'accélération baisse à $250\times$ dans un cas homogène et $120\times$ dans un cas hétérogène. Le cas hétérogène fonctionne en *couches* ce qui rend plus facile l'assignation des paramètres d'intérêt pour une couche lorsque la simulation ne contient qu'un petit nombre de ces dernières. Les paramètres

peuvent être placés en mémoire constante ou en mémoire partagée, donc sans avoir à utiliser la mémoire globale. Cette technique ne peut s'appliquer dans le cas d'un volume de voxels acquis par imagerie CT puisque la gamme des unités Hounsfield acquise, et donc des données de sections efficaces à stocker, dépassera la taille de ces mémoires rapides.

L'implémentation GPU est expliquée sans démontrer d'aspects spécifiques à la programmation sur carte graphique excepté l'utilisation des différents niveaux de mémoire. Ceci peut s'expliquer par le fait que c'est la première implémentation GPU du genre et que le problème de la simulation de Monte Carlo s'adapte facilement à une architecture SIMD. L'architecture de l'implémentation CPU est largement conservée et les résultats numériques sont donc eux aussi comparables.

1.5.2 gDPM

Un autre code portant sur le sujet des simulations Monte Carlo sur GPU est présenté par Jia *et al.* (Jia et al., 2010a). Leur implémentation consiste en un port sur GPU de l'algorithme DPM déjà présenté à la section 1.3. Encore une fois, les particularités informatiques portent surtout sur l'emplacement des données en mémoire. Il s'agit toutefois de l'implémentation la plus proche de ce que représente le travail fait dans cette thèse.

Leur implémentation étant un port d'algorithme existant, plusieurs facettes de l'implémentation sont peu compatibles avec une approche GPU. Premièrement, ils utilisent un seul processus par particule primaire. Lors d'un trajet de particule primaire, plusieurs particules secondaires sont générées et doivent être simulées. Or, en utilisant un seul processus par particule primaire, un nombre variable de particules sera simulé pour chaque processus puisque le nombre de particules secondaires par particule primaire est une variable aléatoire. De plus, découlant encore du fait qu'un seul processus est utilisé, il est probable qu'à l'intérieur d'un même bloc de processus, un processus doive simuler un photon alors qu'un autre doive simuler un électron. Le code pour ces deux simulations étant probablement complètement différent, une forte sérialisation aura lieu.

Les résultats en termes de facteurs d'accélération sont près de 5×. Les auteurs raisonnent que cela est dû à l'importante divergence intrinsèquement liée aux simulations de Monte Carlo ou plusieurs processus d'un même groupe d'exécution (*warp*) prennent des branches différentes d'une section de code conditionnelle. Comme illustré plus haut, leur implémentation non conforme avec une approche par GPU est aussi en cause. Les résultats de doses sont à l'intérieur de l'incertitude statistique lorsque comparés à des résultats de DPM, ce qui est sans surprise étant donné que le même algorithme est utilisé.

1.5.3 Transport de photons par PENELOPE

Un troisième article publié par Badal et Badano (Badal and Badano, 2009) porte sur le transport de photons. Il s'agit en fait d'une note technique très courte qui fait état de leur adaptation de l'algorithme de transport de photons présent dans PENELOPE. De plus, leur programme vise plutôt à modéliser les interactions présentes lors de la prise d'image à rayons X ce qui fait que l'énergie des photons simulés est beaucoup plus basse (<50 keV) que celle employée lors d'un traitement en radiothérapie. Cette faible énergie réduit le nombre d'interactions possibles par particule et donc la divergence en termes de temps de vie de la simulation. Leur implémentation utilise l'algorithme de Woodcock qui a déjà été présenté. Ils obtiennent un résultat de 27× d'accélération en comparaison avec une implémentation CPU équivalente.

1.5.4 Charges de travail hétérogène

Finalement, un article portant sur le rendu graphique de Aila et Laine (Aila and Laine, 2009) a apporté quelques éclaircissements quant au comportement des cartes NVIDIA et des charges de travail déséquilibrées. Leur article se concentre sur les algorithmes de traçage de rayons. Le temps passé à tracer un rayon donné peut être plus ou moins long selon le nombre de surfaces que celui-ci rencontre avant d'être terminé. Ceci peut s'apparenter à la simulation de Monte Carlo où une particule subit un nombre plus ou moins grand d'interactions avant

de quitter la géométrie ou d'être complètement absorbée.

Leur principale découverte est que lors d'une simulation avec une charge de travail débalancée, il est possible de n'être limité ni par la largeur du bus mémoire ni par la puissance de calcul offerte par le matériel. En effet, la différence entre la performance qu'ils obtiennent dans un cas idéal où chaque rayon accède aux mêmes éléments en mémoire, de façon groupée, est de 1.7 à 2.4 fois plus lent que la performance maximale atteignable déduite à l'aide d'un simulateur de l'architecture utilisée. Ils proposent que ceci est dû au fait que certains processus d'un groupe SIMD retiennent le reste du groupe en otage puisque leur charge de travail est plus longue que les autres du groupe. Cette prise en otage d'un des processus fait que les autres du groupe sont en attente et que la puissance de calcul est perdue. De plus, puisque le groupe en entier doit terminer avant que les ressources utilisées par ce groupe de processus ne soit libérées, le prochain groupe de processus ne peut être lancé sur le MP. Ce phénomène est très probable dans une simulation de Monte Carlo si une implémentation naïve lance autant de processus que de particules à simuler et que pour chaque particule il y a une boucle *while* qui ne sort que lorsque la particule atteint un certain niveau d'énergie.

La solution de Aila *et al.* à ce problème est de lancer seulement autant de processus que le matériel peut en supporter. Par exemple, pour une carte avec 30 MP, ceci représenterait 240 processus. Ces processus vont alors chercher leur travail d'une file de rayons ou, dans le cas d'intérêt pour cette thèse, de particules. Lorsque la queue de travail est vide, la simulation est terminée. Ce détournement du modèle traditionnel CUDA où plusieurs milliers de processus sont lancés fait qu'un groupe de processus ne peut tenir un MP en otage puisqu'il n'y a qu'un groupe assigné à chaque MP. Cette approche pourrait aussi rendre possible l'utilisation efficace de la technique de réduction de variance de répétition d'historiques puisqu'un seul bloc est lancé par MP et que ce bloc traitera un grand nombre de particules. Il est donc envisageable de conserver des historiques à répéter en mémoire partagée.

1.6 But et objectifs de la présente étude

Le but principal de cette thèse est l'étude de l'utilisation du processeur graphique (GPU) pour l'implémentation d'une plateforme de calcul de dose par simulation de Monte Carlo dans un contexte de calculs en radiothérapie. L'atteinte de ce but aura pour effet de rendre la simulation de Monte Carlo utilisable dans un contexte clinique où le temps de calcul est un élément critique.

Le premier objectif consiste à l'étude de la plateforme Monte Carlo pour des cas de radiothérapies externes où des énergies particules d'énergie de 10 keV-20MeV sont lancées d'une source polyénergétique externe. Une validation dans un ensemble de cas ainsi qu'une comparaison avec une plateforme de calcul Monte Carlo générale et établie sera effectuée. Chronologiquement, c'est aussi lors de l'atteinte de cet objectif que le comportement des diverses interactions sera établi et où les routines de simulation d'interaction les plus adéquates seront choisies.

Le deuxième objectif consiste en l'étude du comportement de l'engin pour des situations de curiethérapie à bas débit où plusieurs sources de faibles énergies (20-35 keV) sont présentes à l'interne. Comme les sources internes ne peuvent être représentées pratiquement à l'aide de voxels, une méthode paramétrique pour représenter les géométries ainsi qu'une manière efficace de parcourir cette géométrie devra être élaborée. L'atteinte de cet objectif aura pour effet de permettre un calcul de dose rapide utilisant la simulation de Monte Carlo plutôt que l'approche simpliste recommandée dans le rapport du TG-43 (Rivard et al., 2004).

Le troisième objectif consiste en la sélection de la meilleure méthode pour effectuer du calcul de transport d'énergie en présence de champs magnétiques externes ainsi que la validation du calcul avec des mesures expérimentales et une plateforme de calcul générale et établie. Les plateformes Monte Carlo prenant en considération le champ magnétique prennent plusieurs heures, voire même plusieurs jours, pour calculer la dose dans un patient. L'atteinte de cet objectif aura pour effet de rendre possible un calcul de dose, en temps acceptable, pour un patient recevant un traitement dans un appareil hybride comportant une résonance

magnétique (IRM) et un accélérateur linéaire, le MRI-Linac (Lagendijk et al., 2008).

CHAPITRE 2

DÉMARCHE DE L'ENSEMBLE DU TRAVAIL DE RECHERCHE ET ORGANISATION GÉNÉRALE DU DOCUMENT

Pour atteindre les objectifs cette thèse, trois articles scientifiques ont été rédigés. Ce chapitre présente ces trois articles et les positionne à l'intérieur du but global de cette thèse.

2.1 Article 1 `GPUMCD` : Implémentation GPU du calcul de transport d'énergie par simulation Monte Carlo

Cette section porte sur le développement et l'implémentation de `GPUMCD`, le logiciel de simulation de Monte Carlo pour le calcul de la dose présenté dans cette étude. Le but de `GPUMCD` est de permettre l'utilisation de la simulation de Monte Carlo pour des calculs de dose routiniers, ce qui n'est habituellement pas possible dû aux longs temps de calcul associés à cette méthode. `GPUMCD` est développé pour tirer avantage du processeur graphique, une plateforme de calcul hautement parallèle. Cette plateforme est toute indiquée pour le calcul d'une simulation Monte Carlo en transport d'énergie puisque ce calcul est lui aussi hautement parallèle.

2.1.1 Objectifs de l'article scientifique

Ce premier article scientifique a introduit `GPUMCD` à la communauté en présentant la philosophie d'implémentation, le détail des interactions physiques modélisées ainsi que des résultats de comparaisons effectuées avec des fantômes simples et une plateforme de calcul en simulation de Monte Carlo générale et établie : EGSnrc (Kawrakow and Rogers, 2000). Ce premier article de validation se limite à des sources simples et à des fantômes en couches superposées.

Cet article répond au premier objectif de cette thèse, soit : « l'étude de la plateforme Monte

Carlo pour des cas de radiothérapies externes ».

2.1.2 Résultats et impact

Les résultats présentés dans cet article sont regroupés en deux catégories : exactitude et rapidité.

Du point de vue de l'exactitude, la plateforme `GPUMCD` a été comparée à EGSnrc à l'aide de cas de fantômes en couches superposées. L'outil de comparaison du critère gamma a été utilisé (Low and Dempsey, 2003). Ce critère a l'avantage de prendre en considération à la fois la différence dosimétrique mais aussi la différence géométrie en comparant les résultats. Par exemple, dans une région de gradient élevée, un résultat peut être dosimétriquement loin de la valeur de référence, mais géométriquement proche. Une valeur acceptée de critère gamma à respecter pour le calcul de la dose dans un milieu homogène et avec un faisceau de photons externe est de 2%-2mm (Van Dyk et al., 1999). La présente étude a respecté ce critère dans tous les cas, excepté un cas de calcul de faisceau d'électrons en géométrie hétérogène, situation où le critère de 2%-2mm serait considéré strict puisqu'un critère 3%-3mm est habituellement utilisé pour des cas hétérogènes.

Du point de vue de la rapidité d'exécution, `GPUMCD` a été systématiquement plus rapide que EGSnrc mais aussi que DPM, une plateforme développée pour le calcul rapide. Le gain en vitesse et en efficience de calcul a été, pour chaque situation et chaque type de faisceau, d'au moins 250× par rapport à DPM et 1200× par rapport à EGSnrc.

L'impact de cet article est qu'un outil de calcul permettant de faire du calcul de dose routinier à l'aide d'une simulation de Monte Carlo est possible. En effet, l'article fait état d'environ 40 ms/1M particules pour un faisceau de photon de 15 MeV.

2.1.3 Contribution et permissions

Sami Hissoiny

– Idée originale
– Implémentation de `GPUMCD`
– Validation de `GPUMCD` avec EGSnrc
– Écriture de l'article

Hugo Bouchard
– Discussion scientifique
– Correction de l'article

Benoit Ozell
– Codirecteur de projet
– Correction de l'article

Philippe Després
– Codirecteur de projet
– Correction de l'article

2.1.4 Information sur la publication

Cet article scientifique a été publié en 2011 dans la revue *Medical Physics* (Hissoiny et al., 2011a). Cette revue a été choisie puisqu'elle rejoint le lectorat le plus susceptible de trouver cette nouvelle plateforme utile et intéressante.

Un deuxième manuscrit a été écrit mais n'a pas été révisé par les pairs. Ce manuscrit décrit la génération de nombres pseudo-aléatoires sur GPU et, entre autres, le générateur utilisé dans `GPUMCD`. Ce manuscrit a été placé à l'annexe III. Ce travail était requis au début de l'étude puisqu'un générateur adéquat n'existait pas dans la suite CUDA. Ce travail a été rendu moins intéressant depuis la venue d'un générateur de nombres pseudo-aléatoires dans CUDA mais le générateur présenté dans la manuscrit est toujours utilisé dans `GPUMCD`.

2.2 Article 2 GPUMCD : Étude de la précision pour calcul en curiethérapie à bas débit

Cet article porte sur l'évaluation des capacités d'utiliser GPUMCD dans un cadre de calcul en curiethérapie. L'application de la plateforme dans ce contexte n'est pas trivial puisque la plateforme a été développée pour les énergies rencontrées en radiothérapie externe qui sont passablement plus élevées que celles rencontrées en curiethérapie (MeV versus bas keV). Le traitement des événements physiques en basse énergie n'est donc pas aussi complet qu'il pourrait l'être, dans un but d'efficacité et de rapidité pour le calcul en radiothérapie externe. L'omission d'un type d'interaction, l'effet Rayleigh, ainsi que le traitement simpliste de l'effet photoélectrique pourraient compromettre l'exactitude de la simulation.

2.2.1 Objectifs de l'article scientifique

L'objectif principal de cet article est de vérifier l'exactitude avec laquelle GPUMCD est capable de produire des distributions de dose dans un contexte de curiethérapie. Pour ce faire, GPUMCD est utilisé pour produire des paramètres TG-43 (Rivard et al., 2004) pour deux sources de curiethérapie différentes. La comparaison de ces résultats sera faite avec d'autres présents dans la littérature, et qui auront été produits avec des plateformes Monte Carlo plus exactes en basse énergie.

Un autre objectif est de démontrer avec quelle rapidité ces calculs peuvent être faits.

Du point de vue pratique, un nouveau module de traitement des surfaces paramétriques du deuxième degré, qui n'était pas présent dans la première itération, a été implémenté, validé et utilisé aux fins de cet article.

2.2.2 Résultats et impact

Il n'existe pas, dans le rapport du TG-43, de critère d'acceptabilité d'une nouvelle plateforme Monte Carlo pour le calcul en curiethérapie. Il n'est donc pas possible de classer, ou non, `GPUMCD` comment étant assez exacte. Il est par contre possible de rapporter que les résultats produits par `GPUMCD` et les plateformes généralistes sont à l'intérieur de 1% à 4%, selon le paramètre étudié.

Du point de vue de la rapidité, environ 40 ms sont requises pour simuler 1 M de particules primaires.

2.2.3 Contribution et permissions

Sami Hissoiny
- Idée originale
- Implémentation de `GPUMCD` pour curiethérapie
- Validation de `GPUMCD` pour curiethérapie
- Écriture de l'article

Benoit Ozell
- Codirecteur de projet
- Correction de l'article

Philippe Després
- Codirecteur de projet
- Correction de l'article

Jean-François Carrier
- Discussion scientifique
- Correction de l'article

2.2.4 Information sur la publication

Cet article scientifique a été publié en 2011 dans la revue *Medical Physics* (Hissoiny et al., 2011b). Cette revue a été choisie puisque l'article contient des résultats, en termes d'exactitude, utiles à la communauté de physique médicale.

2.3 Article 3 `GPUMCD` : Calculs en présence de champ magnétique

2.3.1 Objectifs de l'article scientifique

Le but de cet article est d'utiliser `GPUMCD` comme plateforme de calcul de dose dans le contexte du MRI-Linac, une modalité d'imagerie-traitement jumelant un imageur par résonance magnétique (IRM) et un accélérateur linéaire. Cette modalité requiert un traitement de dose qui prend en considération l'effet du champ magnétique sur les particules chargées (sur les électrons dans le cas de `GPUMCD`). Seule la méthode de Monte Carlo, à ce jour, peut prendre en considération l'effet du champ magnétique sur les électrons.

Le premier objectif de l'article est de vérifier expérimentalement l'exactitude du formalisme de transport dans un champ magnétique ajouté à `GPUMCD`. Deux expériences, une en rendement en profondeur et l'autre en profil latéral, sont effectuées. Les expériences contiennent des interfaces eau-air, là où les effets du champ magnétique sont les plus prononcés.

Le deuxième objectif est de définir le temps d'exécution nécessaire pour faire un calcul de dose typique dans une géométrie patient et avec un spectre de faisceau réaliste. De plus, l'impact de la prise en considération du champ magnétique sur le temps d'exécution est étudié.

2.3.2 Résultats et impact

Les résultats de validation expérimentale démontrent que `GPUMCD` respecte des critères gamma 2%-2mm versus les données expérimentales et ce pour les deux expériences. Les effets du

champ sont observés aux interfaces eau-air.

La vitesse d'exécution trouvée est de moins de 15 secondes par faisceau pour une précision statistique de 2% (1σ) à l'intérieur d'un champ magnétique de 1.5 T, lorsqu'exécutée sur une GeForce GTX480.

Ce résultat (parmi d'autres dans l'article) montre l'applicabilité de `GPUMCD` pour faire un calcul *en ligne* pour le MRI-Linac. Cette fonctionnalité est essentielle pour permettre le calcul d'un nouveau plan de traitement si, au moment du traitement, une différence au niveau de l'anatomie du patient est trouvée par l'IRM. De façon plus générale, le MRI-Linac requérant la simulation de Monte Carlo pour tous ses calculs de dose, une plateforme étant capable de faire ces calculs dans un temps raisonnable est nécessaire pour une éventuelle utilisation à grande échelle de la modalité d'imagerie-traitement.

2.3.3 Contribution et permissions

Sami Hissoiny
- Idée originale
- Implémentation de l'effet du champ magnétique sur les électrons
- Validation de `GPUMCD` pour le transport d'énergie dans un champ magnétique
- Écriture de l'article

Alexander Raaijmakers
- Mesures expérimentales
- Génération de l'espace de phase
- Correction de l'article

Benoit Ozell
- Codirecteur de projet
- Correction de l'article

Philippe Després
- Codirecteur de projet

– Correction de l'article

Bas Raaymakers

– Idée originale

– Discussion scientifique

– Correction de l'article

2.3.4 Information sur la publication

Cet article est publié dans la revue « Physics in Medicine and Biology » (Hissoiny et al., 2011c). Cette revue a été choisie puisque c'est dans celle-ci que tous les autres travaux sur le MRI-Linac de l'équipe de Bas Raaymakers ont été publiés.

CHAPITRE 3

GPUMCD : A NEW GPU-ORIENTED MONTE CARLO DOSE CALCULATION PLATFORM

Sami Hissoiny

École polytechnique de Montréal, Département de génie informatique et génie logiciel, 2500 chemin de Polytechnique. Montréal (Québec), CANADA H3T 1J4

Hugo Bouchard

Département de Physique, Université de Montréal, Pavillon Roger-Gaudry (D-428), 2900 Boulevard Édouard-Montpetit, Montréal, Québec H3T 1J4, Canada and Département de Radio-oncologie, Centre hospitalier de l'Université de Montréal (CHUM), 1560 Sherbrooke est, Montréal, Québec H2L 4M1, Canada

Benoît Ozell

École polytechnique de Montréal, Département de génie informatique et génie logiciel, 2500 chemin de Polytechnique. Montréal (Québec), CANADA H3T 1J4

Philippe Després

Département de Physique, Université de Montréal, Pavillon Roger-Gaudry (D-428), 2900 Boulevard Édouard-Montpetit, Montréal, Québec H3T 1J4, Canada and Département de Radio-oncologie, Centre hospitalier de l'Université de Montréal (CHUM), 1560 Sherbrooke est, Montréal, Québec H2L 4M1, Canada

Abstract

Purpose : Monte Carlo methods are considered the gold standard for dosimetric computations in radiotherapy. Their execution time is however still an obstacle to the routine use

of Monte Carlo packages in a clinical setting. To address this problem, a completely new, and designed from the ground up for the GPU, Monte Carlo dose calculation package for voxelized geometries is proposed: GPUMCD.

Method : GPUMCD implements a coupled photon-electron Monte Carlo simulation for energies in the range 0.01 MeV to 20 MeV. An analogue simulation of photon interactions is used and a Class II condensed history method has been implemented for the simulation of electrons. A new GPU random number generator, some divergence reduction methods as well as other optimization strategies are also described.

Results : Dosimetric results obtained with GPUMCD were compared to EGSnrc. In all but one test case, 98% or more of all significant voxels passed a gamma criteria of 2%-2mm. In terms of execution speed and efficiency, GPUMCD is more than 1200 times faster than EGSnrc and 250 times faster than DPM, a Monte Carlo package aiming fast executions. Absolute execution times of less than 0.3 s are found for the simulation of 1M electrons and 4M photons in water.

Conclusion : GPUMCD, a new GPU-oriented Monte Carlo dose calculation platform, has been compared to EGSnrc and DPM in terms of dosimetric results and execution speed. Its accuracy and speed make it an interesting solution for full Monte Carlo dose calculation in radiation oncology.

3.1 Introduction

Monte Carlo packages such as EGS4 (Nelson et al., 1985), EGSnrc (Kawrakow and Rogers, 2000; Kawrakow and Rogers, 2003), EGS5 (Hirayama et al., 2005) and PENELOPE (Salvat et al., 2006) have been extensively compared to experimental data in a large array of conditions, and generally demonstrate excellent agreement with measurements. EGSnrc, for instance, was shown to reproduce experimental data within 0.1% (Rogers, 2006). Despite this accuracy, Monte Carlo platforms are mostly absent from routine dose calculation in radiation therapy due to the long computation time required to achieve sufficient statistical significance.

Monte Carlo simulations are generally considered the gold standard for benchmarking analytical dose calculation approaches such as pencil-beam based computations (Mohan et al., 1986; Ahnesjö et al., 1992; Jeleń et al., 2005) and convolution-superposition techniques (Mackie et al., 1985; Liu et al., 1997). Such techniques typically use precomputed Monte Carlo data to incorporate physics-rich elements in the dose calculation process. These semi-empirical methods were developed as a trade-off between accuracy and computation time and as such do not match the level of accuracy offered by Monte Carlo simulations, especially in complex heterogeneous geometries where differences of up to 10% were reported (Krieger and Sauer, 2005; Ding et al., 2005; Ma et al., 2000).

Fast Monte Carlo platforms have been developed for the specific purpose of radiotherapy dose calculations, namely VMC (Kawrakow et al., 1996; Fippel, 1999; Gardner et al., 2007) and DPM (Sempau et al., 2000). These packages make some sacrifices to the generality and absolute accuracy of the simulation by, for example, restricting the energy range of incoming particles. This assumption allows simpler treatment of certain particle interactions which are less relevant within the limited energy range considered. Gains in efficiency of around 50 times can be achieved by these packages when compared to general-purpose solutions.

This paper presents GPUMCD (GPU Monte Carlo Dose), a new code that follows the fast Monte Carlo approach. GPUMCD relies on a relatively new hardware platform for computations: the GPU (Graphics Processing Unit). The GPU is gaining momentum in medical physics (Kutter et al., 2009b) (Després et al., 2008) (de Greef et al., 2009) (Hissoiny et al., 2010a), as well as in other spheres (Satish et al., 2009) (Micikevicius, 2009) (Thomas et al., 2009) (Volkov and Demmel, 2008) (Wang et al., 2009) where high-performance computing is required. A layer-oriented simulation based on the MCML Monte Carlo code for photon transport has already been ported to the GPU (Alerstam et al., 2008). The photon transport algorithm from PENELOPE has also been ported to the GPU (Badal and Badano, 2009) for accelerations of up to 27x. DPM (Sempau et al., 2000) has been ported to the GPU (Jia et al., 2010a) with excellent dosimetric results compared to the CPU version, as expected, but with relatively low (5-6.6x) acceleration factors.

A comparison of GPUMCD to the work by Jia *et al.* must be made cautiously. The work of Jia *et al.* is a port of an existing algorithm that is not, to the best of our knowledge, designed for the specific nature of the GPU. Notable differences are found between the two approaches regarding the treatment of secondary particles and the electron-photon coupling. These differences will be exposed in Sec. 3.2.3.3. The code presented in this article is the first attempt of a complete rewrite of a coupled photon-electron Monte Carlo code specifically designed for the GPU. New techniques for the memory management of particles as well as efforts to reduce the inherent divergent nature of Monte Carlo algorithms are detailed. The physics behind the platform have been carried out with respect to the literature; all theoretical developments of cross section sampling methods have been adapted from their description in general-purpose Monte Carlo package manuals (and references therein) listed previously. However, their actual implementation is original, as it is specifically tailored for parallel execution on graphics hardware, and the code has not been taken from existing Monte Carlo implementations.

The paper is structured as follows: Sec. 5.2 introduces the physics principles as well as the hardware platform and implementation details. Sec. 5.3 presents the results in terms of dose and calculation efficiency of GPUMCD compared to the EGSnrc and DPM platforms. Finally, a discussion of the reported results is presented in Sec. 3.4.

3.2 Material and methods

3.2.1 Photons simulation

Photon transport can be modeled with an analogue simulation, *i.e.* that every interaction is modeled independently until the particle leaves the geometry or its energy falls below a certain energy level referenced as P_{cut} from now on. This allows for an easy implementation of the transport process.

GPUMCD takes into account Compton scattering, the photoelectric effect and pair produc-

tion. Rayleigh scattering can be safely ignored for the energy range considered (10 keV-20 MeV). Considering a homogeneous phantom, the distance to the next interaction, noted s, can be sampled from the following expression:

$$s = -\frac{1}{\mu(E)} ln(\zeta), \tag{3.1}$$

where ζ is a random variable uniformly distributed in [0,1] and $\mu(E)$ is the value of the total attenuation coefficient, given by

$$\mu = \frac{N_A}{A(\vec{x})} \rho(\vec{x}) \left(\sigma_{compton} + \sigma_{photo} + \sigma_{pair}\right), \tag{3.2}$$

where the σ's are total cross section values for the corresponding interactions, N_A is Avogadro's number, A is the molecular weight and ρ is the density.

To eliminate the need for a distinct geometry engine in heterogeneous situations, GPUMCD employs the Woodcock raytracing algorithm (Woodcock et al., 1965) in which the volume can be considered homogeneously attenuating by introducing a fictitious attenuation coefficient, corresponding to a fictitious interaction which leaves the direction and energy of the particle unchanged, in every voxel $\vec{x}$:

$$\mu(\vec{x})_{fict} = \mu_{max} - \mu(\vec{x}) \tag{3.3}$$

where μ_{max} is the attenuation coefficient of the most attenuating voxel inside the volume. The distance to the next interaction is therefore always sampled using μ_{max}. The sampling of the interaction type, once at the interaction position, is then performed taking into consideration this fictitious interaction type. This method is not an approximation, it leads to the correct result even in arbitrarily heterogeneous geometries.

Compton scattering is modeled with a free electron approximation using the Klein-Nishina cross section (Klein and Nishina, 1929). The energy and direction of the scattered photon and secondary electron are sampled according to the direct method derived by Everett *et*

al. (Everett et al., 1971). Binding effects and atomic relaxation are not modeled. These approximations are reasonable when the particle energy is large compared to the electron binding energy which is the case for radiation therapy (Johns and Cunningham, 1983).

The photoelectric effect is modeled by once again ignoring atomic relaxation. Additionally, shell sampling is also ignored and all electrons are assumed to be ejected from the K-shell. The sole product of the interaction is a new electron in motion with total energy $E_{elec} = E_{phot}$. The angular sampling of the electron is selected according to the Sauter distribution (Sauter, 1931) following the sampling formula presented in Sec. 2.2 of the PENELOPE manual (Salvat et al., 2006).

Relatively crude approximations are employed when simulating pair production events. First and foremost, no positrons are simulated in this package. The pair production event instead generates two secondary electrons. The relatively low probability of pair production events at the energies considered as well as the additional low probability of in-flight annihilation of the positron make this approximation suitable (Sempau et al., 2000). Triplet production is also not modeled. The energy of the incoming photon is trivially split between the two electrons using a uniformly distributed random number. The angle of both electron is sampled using the algorithm presented in Eq. 2.1.18 of the EGSnrc manual (Kawrakow and Rogers, 2003).

3.2.2 Electrons simulation

Because of the much larger number of interactions experienced by a charged particle before it has deposited all of its energy, analogue simulations cannot be used practically. A class II condensed history method is used here, as defined by Berger (Berger, 1963), in which interactions are simulated explicitly only if they cause a change in orientation greater than θ_c or a change in energy larger than E_c. Below these thresholds, the interactions are modeled as taking part of a larger condensed interaction, the result of which is a major change in direction and energy. Above the thresholds, interactions, usually called hard or catastrophic interactions, are modeled in an analogue manner similar to the simulation of photons.

3.2.2.1 Hard interactions

GPUMCD simulates inelastic collisions as well as bremsstrahlung in an analogue manner, as long as the changes to the state of the particle are greater than the previously discussed threshold.

For inelastic collisions, a free electron approximation is used and no electron impact ionization nor spin effects are modeled. The Møller cross section is used to describe the change in energy and direction of the scattered and knock-on electrons. The sampling routine presented in Sec. 2.4.3.i of the EGSnrc manual (Kawrakow and Rogers, 2003) is used.

During a bremsstrahlung event, the energy of the created photon is sampled as presented in Sec. 2.4.2.ii of the EGSnrc manual. Approximations to the angular events are employed : the angle of the electron is unchanged and the angle of the photon is selected as $\theta = m_0c^2/E$ where E is the energy of the incoming electron.

3.2.2.2 Multiple scattering

The class II condensed history method requires the selection of an angle for the multiply-scattered electron after a step, as discussed in the next subsection. To this end, the formulation presented by Kawrakow and Bielajew (Kawrakow and Bielajew, 1998) and Kawrakow (Kawrakow and Rogers, 2000) is used. The required q^{2+}_{SR} data are imported from the `msnew.dat` file packaged with EGSnrc.

3.2.2.3 Electrons transport

The transport of electrons is less straightforward than photon transport because of the high probability of soft collisions, which are regrouped in a single step between two hard interactions (collision or bremsstrahlung events). This condensed history method introduces an unphysical (but mathematically converging) factor for the length of the multiple-scattering

step. Between consecutive hard collisions, which are separated by a distance sampled with an electron version of Eq. (3.1), a number of multiple-scattering events will occur. In GPUMCD, a simple step-length selection (henceforth $|e_s|$) is used and can be summarized as

$$|e_s| = min\{d_{vox}, |e_s|_{max}, |e_{hard}|\} \tag{3.4}$$

where d_{vox} is the distance to the next voxel boundary, $|e_s|_{max}$ is a user-supplied upper bound of step length and $|e_{hard}|$ is the distance to the next hard interaction. It is probably a weak point of this platform that will be addressed in future releases.

The distance to the next hard interaction can once again be sampled with the help of the Woodcock raytracing algorithm. For electron transport, however, the total attenuation coefficient, $\mu(E)$, is in fact $\mu(E, s)$ where the dependence on distance has been made explicit. The dependence on distance is due to the fact that electrons will lose energy due to sub-threshold interactions between two successive hard interactions. In this application, the approximation is made that the attenuation coefficient is decreasing with respect to the particle energy. It is not rigorously true, as shown in (Kawrakow and Rogers, 2000), but allows for an easier treatment since $\mu_{max}(E)$ can be selected with E being the energy at the beginning of the electron step.

Energy losses are experienced by the electron during the soft scattering events that are modeled by the multiple-scattering step. This energy loss is accounted for with the continuously slowing down approximation (CSDA). This approximation states that charged particles are continuously being slowed down due to the interactions they go through and that the rate of energy loss is equal to the stopping power S, given by:

$$S(E) = -\frac{dE}{ds}. \tag{3.5}$$

The implementation of a class II condensed history method only accounts for sub-threshold

interactions in the CSDA. The use of restricted stopping powers, L, is therefore required:

$$L(E, K_{cut}) = \int_0^{K_{cut}} \Sigma(E, E')dE' \tag{3.6}$$

where $\Sigma(E, E')$ is the total macroscopic cross section. Since this release of GPUMCD is $|e_s|$-centric, the energy loss of a given step is computed with

$$\Delta E = \int_0^{|e_s|} L(s)ds. \tag{3.7}$$

The evaluation of the integral can be reduced to

$$\Delta E = L\left(E_0 - L(E_0)s/2\right) \tag{3.8}$$

which is EGSnrc's Eq. 4.11.3.

Between two consecutive hard collisions, the electron does not follow a straight line. Bielajew's alternate random hinge method (Salvat et al., 2006) builds on PENELOPE's random hinge method to handle the angular deflection and lateral displacement experienced by an electron during an electron step. The random hinge method can be described as follows: the electron is first transported by $\zeta|e_s|$ (where ζ is again a uniformly distributed random number), at which point the electron is rotated by the sampled multiple- scattering angle described in Sec. 3.2.2.2 and the energy loss is deposited. The electron is then transported for the remaining distance equal to $(\zeta - 1)|e_s|$. Bielajew's refinement of the algorithm involves randomly sampling the angular deflection either before or after the electron has deposited the energy associated with the step.

3.2.3 GPU implementation

In this section, different aspects of the GPU implementation are detailed. GPUMCD is built with the CUDA framework from NVIDIA. A general description of the GPU building blocks and memory levels can be found in (Hissoiny et al., 2009). Sec. 3.2.3.1 presents the random

number generator used in this work while Sec. 3.2.3.2 describes the memory management. Finally, Sec. 3.2.3.3 addresses the SIMD divergence issue and presents solutions as well as other optimizations used in GPUMCD.

3.2.3.1 Random number generator

A pseudo random number generator (PRNG) is already available in the NVIDIA SDK (Software Development Kit) for GPU computing. However, it uses a lot of resources which would then be unavailable for the rest of the Monte Carlo simulation. For this reason, a new lightweight PRNG based on the work of Marsaglia (Marsaglia and Zaman, 1991) was implemented. We use a combined multiply-with-carry (MWC) generator, with recurrence of the form

$$x_{n+1} = (a * x_n + c_n) \bmod (b) \tag{3.9}$$

where a is the multiplier and b the base. The carry, c, is defined by:

$$c_{n+1} = \lfloor \frac{(a * x_n + c_n)}{b} \rfloor. \tag{3.10}$$

The carry c is naturally computed with integer arithmetic and bases b of 2^{32} or 2^{16}. The choice of the multiplier a is not arbitrary: the multiplier is chosen so that $ab-1$ is safeprime. The period of the PRNG with such a multiplier is on the order of $(ab-1)/2$. This PRNG has the advantage of using a small amount of memory. By combining two of these generators for each thread, the PRNG uses two 32 bits values for the multipliers and two 32 bits values for the current state. Ten integer operations (3 $shifts$, 2 $ands$, 2 $mults$ and 3 $adds$) are required to generate one new number. It has been shown to pass all tests but one of the TestU01 suite (L'ecuyer and Simard, 2007), a program designed to test the quality of pseudo random numbers, and to achieve 96.7% of the peak integer bandwidth.

3.2.3.2 Memory management

GPUMCD is composed of four main arrays: a list of electrons and a list of photons, an array to store composition identifiers and another for density values corresponding to a numerical phantom. During the initialization phase, a call to the `initParticle` function is made which fills the correct particle array (depending on the source particle type) and according to the source type (*e.g.* with a parallel beam source all particles have a direction vector of (0,-1,0)). The particle arrays are allocated only once, with size `MAXSIZE`. The choice of `MAXSIZE` is graphics card dependent as the particle arrays occupy the better part of the global memory. For instance, on a 1 Gb card, `MAXSIZE` could be set to 2^{20}. Only a fraction of the particle array will be filled with source particles, to leave room for secondary particle creation. If for instance `MAXSIZE`/d particles are generated, then each primary particle can generate an average of d secondary particle without running out of memory. The choice of d is therefore important; it is energy dependent and to a lesser extent geometry dependent. For example, with a monoenergetic 15 MeV beam of photons on a 32 cm deep geometry, a value of $d = 8$ was found to be sufficient to accommodate all secondary particles. A global counter of active particles is kept for both electrons and photons and every time a new particle is added the counter is atomically incremented until it is equal to `MAXSIZE` after which secondary particles are discarded. If the choice of d and `MAXSIZE` are such that the number of particles generated is lower than `MAXSIZE`, then no particles are discarded. Atomic operations are not ideal on parallel hardware but these two integer values, the current number of particles of both types, should be efficiently cached on GF100 class hardware. After the initialization phase is completed, the simulation starts by calling the function to simulate the initial particle type. Subsequently, secondary particles of the other type are generated and put inside the particle array for that type. The other type of particle will then be simulated, and so on until a relatively low number of particle is left in the stack. This number is user-selectable and could be set to 0 to simulate every particle. Since a simulation pass always comes with some overhead, it could also be set to a non-zero value to avoid the launching of a simulation pass with few particles to simulate which could have a negligible impact on the final distribution. If this impact is in fact negligible of course depends on the threshold selected. Photons and

electrons are never simulated in parallel but instead the particles of the *other* type are placed in their respective array and wait there until the array with the *current* type of particle is exhausted. This, in turn, eliminates the divergence due to the photon-electron coupling. Every time a call to a simulation function for electrons or photons is issued, one can consider that every particle of that type is simulated and that the counter for that type of particle is reset to 0. Sec. 3.2.3.3 will show some refinements related to the management of particles.

The other two main arrays, one of composition identifier and one of density values, are stored as 3D textures since they are used to represent the volume of voxels. Every time a particle is moved, the composition and density of the current particle voxel are fetched from the 3D textures.

Cross section and stopping power data are stored in global memory in the form of a 1D texture. The cross section and stopping power data are preinterpolated on a regular grid with a value at every 1 keV.

The shared memory usage is relatively limited in this application. For the electrons simulation, some specific composition attributes are required (*e.g.* Z_V, Z_G, etc., in Tab. 2.1 of EGS5) and are stored in shared memory. These data values have not been placed in constant memory since that type of memory is best used when all threads of a warp access the same address, which cannot be guaranteed here since these composition attributes are material dependent. Two particles in different materials would therefore request the constant memory at two different addresses, resulting in serialization. No other use for shared memory has been found since the application is by nature stochastic. For instance, we cannot store a portion of the volume since there is no way to know where all the particles of one thread block will be. Similarly, we cannot store a portion of the cross-section data since there is no way to know in which material and with which energy all the particles of one thread block will be.

3.2.3.3 Divergence reduction and other optimization strategies

Every multiprocessor (MP) inside the GPU is in fact a SIMD processor and the SIMD coherency has to be kept at the warp level, *i.e.* 32 threads. The Monte Carlo simulation being inherently stochastic, it is impossible to predict which path a given particle will take and it is therefore impossible to regroup particle with the same fate into the same warps. When a warp is divergent, it is split into as many subwarps as there are execution paths, leading to a performance penalty. Divergence can be seen when, *e.g.* two particles of the same warp do not require the same number of interactions before exhaustion or do not interact in the same way. Some software mechanisms can be employed to reduce the impact of divergence.

The first mechanism consists in performing a stream compaction after N simulation steps, where N is user-defined and corresponds to a number of interactions for photons and to a number of catastrophic events for electrons. For example, if one particle requires 20 interactions to complete and the others in the warp require less than 5, then some scalar processors (SP) in the MP will be idle while they wait for the *slow* particle to finish. This first mechanism then artificially limits the number of simulation steps a particle can undergo during one pass of the algorithm. After this number of steps, every particle that has been completely simulated is removed from the list of particles and the simulation is restarted for another N steps, until every particle has been completely simulated. The removal of particles is not free and therefore it is not clear if this technique will have a positive effect on execution time. The stream compaction is accomplished with the CHAG library (Billeter et al., 2009).

A second mechanism, named persistent thread by its creators and *pool* in GPUMCD, is taken from the world of graphics computing (Aila and Laine, 2009). In this approach, the minimum number of threads to saturate the GPU is launched. These threads then select their workload from a global queue of particles to be simulated. Once a thread is done with one particle, it selects the next particle, until all have been simulated. This is once again to reduce the impact of the imbalance in the number of interactions per particle.

All the GPU code uses single precision floating point numbers as they offer a significant

speed improvement over double precision floating point numbers on graphics hardware. The work of Jia et al. (Jia et al., 2010a) showed that no floating point arithmetic artifacts were introduced for this type of application.

A notable difference between this work and the work by Jia *et al.* is the way the secondary particles are treated. In the work by Jia, a thread is responsible for its primary particle as well as every secondary particles it creates. This can be a major source of divergence because of the varying number of secondary particles created. On the other hand, GPUMCD does not immediately simulate secondary particles but instead places them in their respective particle arrays. After a given pass of the simulation is over, the arrays are checked for newly created secondary particles and if secondary particles are found, they are simulated. This eliminates the divergence due to the different number of secondary particles per primary particles. Additionally, since Jia *et al.* use a single thread per primary particles, a thread may be responsible for the simulation of both electrons and photons which can in turn be a major source of divergence. For example, at time t, thread A may be simulating a secondary electron while thread B a secondary photon. In other words, they simulate both photons and electrons at the same time. Since both of these particles most likely have completely different code path, heavy serialization will occur. On the other hand, since GPUMCD stores secondary particles in arrays to be simulated later, no such divergence due to the electron-photon coupling occurs.

Finally, GPUMCD can be configured to take advantage of a multi-GPU system. The multi-GPU approach is trivial for Monte Carlo simulations: both GPUs execute the `initParticle` function and simulate their own set of particles. After the simulations, the two resulting dose arrays are summed. Linear performance gains are expected for simulations when there are enough particles to simulate to overcome the overhead penalty resulting from copying input data to two graphics card instead of one.

These optimization mechanisms can be turned on or off at the source code level in GPUMCD.

3.2.4 Performance evaluation

The performance evaluation of GPUMCD is twofold: a dosimetric evaluation against EGSnrc and DOSXYZnrc and an efficiency evaluation against EGSnrc and DPM, the later being a fairer comparison since DPM was designed for the same reason GPUMCD was developed, *i.e.* fast MC dose calculations.

All settings for DPM were set to default, notably K_{cut} = 200 keV and P_{cut} = 50 keV. Most parameters for DOSXYZnrz (unless otherwise noted) were also set to default, notably `ESTEPE`=0.25, ξ_{max}=0.5. In EGSnrc/DOSXYZnrc, the boundary crossing algorithm used was `PRESTA-I` and the electron step algorithm was `PRESTA-II`, atomic relaxation was turned off as well as bound Compton scattering. Values of P_{cut} and E_{cut} were set to 0.01 MeV and 0.7 MeV respectively. The same 64^3 grid with 0.5 cm^3 voxels is used for all simulations on all platforms. These values of spatial resolution and number of voxels are insufficient for a clinical calculation with patient specific data; however, they were judged adequate for the benchmarking study conducted here. Preliminary results suggest that no loss of efficiency occurs between GPUMCD and other platforms as the number of voxels is increased.

For dosimetric comparisons, the uncertainty of Monte Carlo simulation results varies as $1/\sqrt{N}$ where N is the number of histories. All simulations ran for visualization purposes on GPUMCD were conducted with enough primary particles to achieve a statistical uncertainty of 1% or less. The same number of particles were then generated in DOSXYZnrc. The graphs are shown without error bars since they would simply add clutter to the results. For execution time comparisons, 4 million particles were generated and simulated for photon beams and 1 million particles for electron beams. All sources were modeled as a monoenergetic parallel beam. All dose distributions were normalized with respect to D_{max}. The efficiency measure, ϵ, was used to evaluate the performance of the different implementations:

$$\epsilon = \frac{1}{s^2 T}, \tag{3.11}$$

where s is the statistical uncertainty and T the computation time. Only voxels with a dose

higher than $0.2 \cdot D_{max}$ were considered for the statistical uncertainty, yielding the following expression for s^2:

$$s^2 = \frac{1}{N_{0.2}} \sum_{D_{ijk}>0.2D_{max}}^{N_{0.2}} \frac{\Delta D_{ijk}}{D_{ijk}}, \tag{3.12}$$

where

$$\Delta D_{ijk} = \sqrt{\frac{\langle D_{ijk}^2 \rangle - \langle D_{ijk} \rangle^2}{N-1}}, \tag{3.13}$$

for N the number of histories simulated and $\langle D \rangle$ the mean of the random variable D.

For the dosimetric evaluation, a gamma index was used to compare the two dose distributions (Low and Dempsey, 2003). The gamma index for one voxel x is defined as

$$\gamma(x) = min\{\Gamma(x, x')\} \forall \{x'\}, \tag{3.14}$$

and

$$\Gamma(x, x') = \sqrt{\frac{||x' - x||^2}{\Delta d^2} + \frac{|D(x) - D(x')|^2}{\Delta D^2}}, \tag{3.15}$$

where $||x' - x||$ is the distance between voxels x and x', $D(x)$ is the dose value of voxel x, Δd is the distance tolerance value and ΔD is the dose tolerance value. The criteria for acceptable calculation is defined for a gamma index value below or equal to 1.0.

All simulations were run on the same PC comprising an Intel Xeon Q6600 and a NVIDIA GeForce GTX480 graphics card. DPM and DOSXYZnrc do not natively support multi-core architectures and have not been modified. GPUMCD, unless running in multi-GPU configuration, uses only one processor core. The CPU used in this study is significantly older than the GPU card used; acceleration numbers should therefore be interpreted accordingly.

3.3 Results

Several slab-geometry phantoms are described in Sec. 3.3.1 . In Sec. 3.3.2, the execution times and gains in efficiency are reported. A multi-GPU implementation is also tested and corresponding acceleration factors are presented.

3.3.1 Dosimetric results

For slab geometries, central axis percent depth dose (PDD) curves as well as dose profiles at different depths or isodose curves are presented. Gamma indices are calculated in the entire volume of voxels and the maximum and average values are reported. The number of voxels with a gamma value higher than 1.0 and 1.2 are also detailed in order to evaluate discrepancies in dose calculation. The gamma criteria used is set to 2% and 2 mm which is a generally accepted criteria for clinical dose calculation (Van Dyk et al., 1999). In every gamma index summary table: 1) γ^{max} is the maximum gamma value for the entire volume of voxel; 2) γ_c^{avg} is the average gamma value for voxels with $D > cD_{max}$; 3) ΣD_c is the number of voxels with $D > cD_{max}$; 4) $\Sigma\gamma_c > g$ is the number (proportion) of voxels with $D > cD_{max}$ and $\gamma > g$.

The first slab geometry is a simple water phantom. GPUMCD treats homogeneous phantoms as heterogeneous phantoms; the particles are transported as if evolving inside a heterogeneous environment and therefore there is no gain in efficiency resulting from homogeneous media. For this geometry, the results for an electron beam and a photon beam are presented in Fig. 3.1 and gamma results in Tab. 3.1.

Table 3.1 Gamma criteria summary for a 15 MeV beam on water.

Particles	γ^{max}	$\gamma_{0.2}^{avg}$	$\Sigma D_{0.2}$	$\Sigma\gamma_{0.2} > 1.0$	$\Sigma\gamma_{0.2} > 1.2$
Electrons	1.24	0.29	1750	37 (2%)	2 (∼0%)
Photons	1.27	0.18	7500	143 (2%)	19 (∼0%)

The second geometry is composed of a lung box inside a volume of water. It is designed to demonstrate the performance of GPUMCD with lateral and longitudinal disequilibrium conditions. For electrons, the lung box is 4 cm long, 4.5 cm wide and starts at a depth of 3 cm; for photons the lung box is 6.5 cm long, 4.5 cm wide and starts at a depth of 5.5 cm. In both cases, the box is centered on the central axis. For this geometry, PDD and isodose plots are presented in Fig. 3.2 for the electron and photon beams while the gamma results are presented in Tab. 3.2.

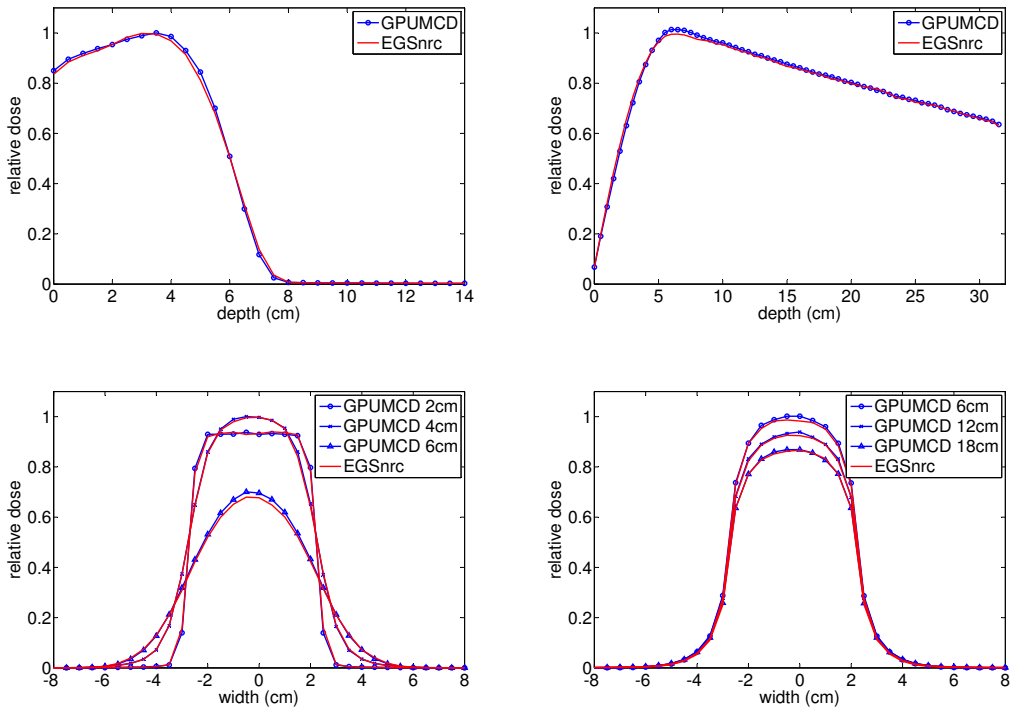

Figure 3.1 PDD (top) and profile (bottom) of 15 MeV electron (left) and photon (right) beams on water.

Table 3.2 Gamma criteria summary for a 15 MeV beam on water with a lung box.

Particles	γ^{max}	$\gamma_{0.2}^{avg}$	$\Sigma D_{0.2}$	$\Sigma\gamma_{0.2} > 1.0$	$\Sigma\gamma_{0.2} > 1.2$
Electrons	1.40	0.32	2366	46 (2%)	2 (0%)
Photons	1.30	0.19	7522	122 (2%)	19 (∼0%)

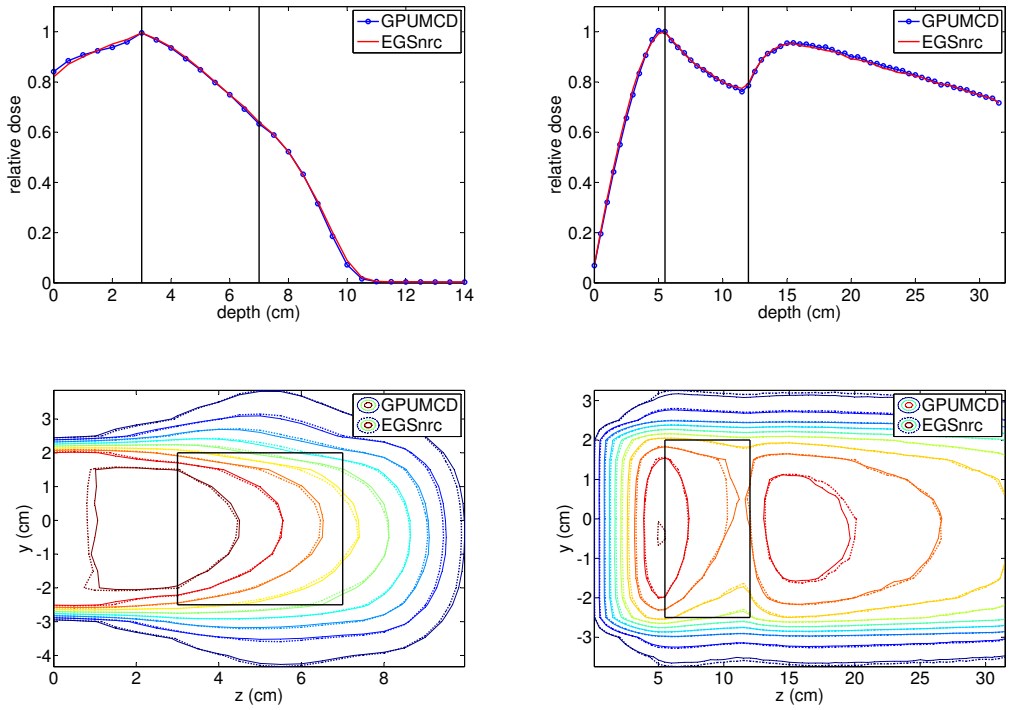

Figure 3.2 PDD (top) and isodose (bottom) of a (left) 15 MeV electron beam on water with a 4 cm long, 4.5 cm wide box of lung at a depth of 3 cm and (right) 15 MeV photon beam on water with a 6.5 cm long, 4.5 cm wide box of lung at a depth of 5.5 cm.

The third geometry is composed of soft tissue, bone and lung. The slabs, for electrons, are arranged as such: 2 cm of soft tissue, 2 cm of lung, 2 cm of bone followed by soft tissue. For photons the geometry is: 3 cm of soft tissue, 2 cm of lung, 7 cm of bone followed by soft tissue. The results for the electron and photon beams are presented in Fig. 3.3 and gamma results in Tab. 3.3.

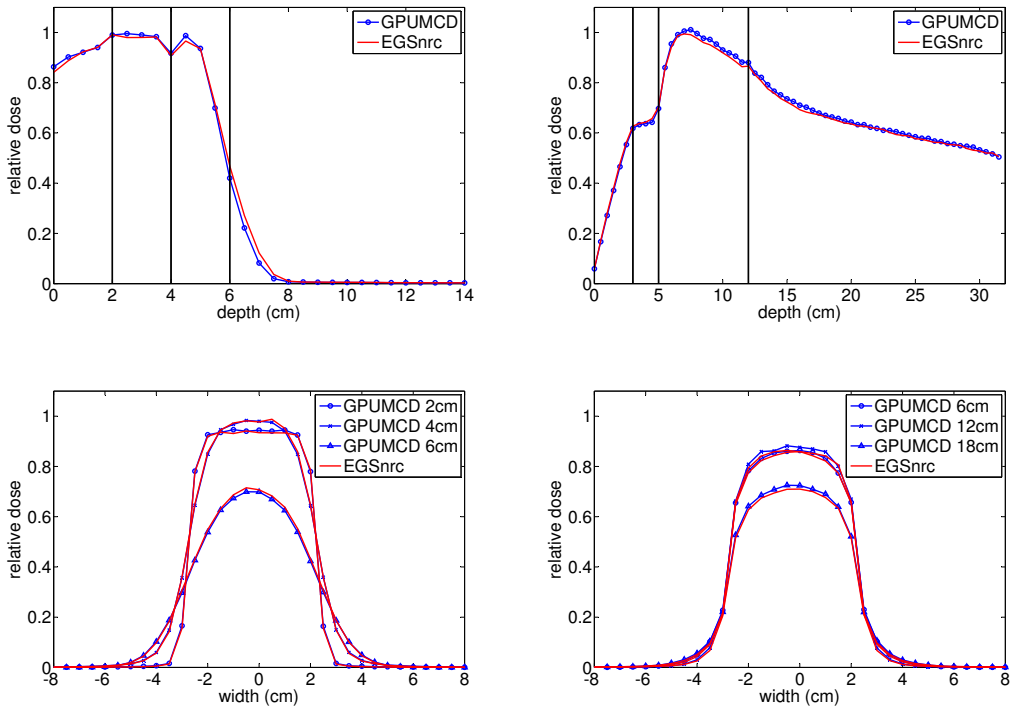

Figure 3.3 PDD (top) and profile (bottom) of a (left) 15 MeV electron beam on a phantom composed of 2 cm of soft tissue, 2 cm of bone, 2 cm of lung followed by soft tissue and (right) 15 MeV photon beam on a phantom composed of 3 cm of soft tissue, 2 cm of lung, 7 cm of bone followed by soft tissue.

3.3.2 Execution time and efficiency gain

In this section, the overall speed and efficiency of GPUMCD is evaluated and compared to EGSnrc and DPM. In the following tables, $Geom_1$ is the water phantom, $Geom_2$ is the water-lung phantom and $Geom_3$ is the tissue-lung-bone phantom. T_{EGSnrc} and T_{DPM} are respectively the EGSnrc and DPM execution times. For the efficiency measurement,

Table 3.3 Gamma criteria summary for a 15 MeV beam on a phantom with layers of soft tissue, bone and lung.

Particles	γ^{max}	$\gamma_{0.2}^{avg}$	$\Sigma D_{0.2}$	$\Sigma\gamma_{0.2} > 1.0$	$\Sigma\gamma_{0.2} > 1.2$
Electrons	1.84	0.34	1700	138 (8%)	54 (3%)
Photons	1.31	0.22	7220	12 (∼0%)	0 (0%)

ϵ, the fastest execution time of GPUMCD (with or without the optimization presented in Sec. 3.2.3.3) has been used. The acceleration factor and efficiency improvement, namely A and A_ϵ, also use the fastest time for GPUMCD. The acceleration factor is defined as

$$A = \frac{T_{ref}}{T_{GPUMCD}}, \tag{3.16}$$

and the efficiency improvement as

$$A_\epsilon = \frac{\epsilon_{GPUMCD}}{\epsilon_{ref}}. \tag{3.17}$$

Tab. 3.4 presents the acceleration results for photon and electron beams where 4M photons and 1M electrons are simulated. Absolute execution times of 0.269 s, 0.275 s and 0.366 s were found with GPUMCD for $Geom_1$, $Geom_2$ and $Geom_3$ respectively for a photon beam and 0.118 s, 0.147 s and 0.162 s for an electron beam.

Table 3.4 Acceleration factors for 1M 15 MeV electrons and 4M 15 MeV photons on the three geometries presented. Absolute execution times of 0.269 s, 0.275 s and 0.366 s were found with GPUMCD for $Geom_1$, $Geom_2$ and $Geom_3$ respectively for a photon beam and 0.118 s, 0.147 s and 0.162 s for an electron beam.

		Photons		Electrons	
		DPM	EGSnrc	DPM	EGSnrc
$Geom_1$	A	584	1498	323	1983
	A_ϵ	665	1515	382	2285
$Geom_2$	A	559	1365	276	1659
	A_ϵ	612	1685	312	1843
$Geom_3$	A	262	1222	296	1617
	A_ϵ	284	1356	318	1785

Tab. 3.5 presents the results of the divergence reduction methods and acceleration strategies presented in Sec. 3.2.3.3 for electron and photon beams. In the following table A_{comp}, A_{pool} represent the acceleration factors with respect to the base configuration execution time (T_{base}), with the stream compaction or *pool* divergence reduction methods enabled, respectively.

Table 3.5 Acceleration results of the different strategies presented in Sec. 3.2.3.3 for the second geometry ($geom_2$).

Particles	T_{base}	A_{comp}	A_{pool}
Electrons	0.293	1.06	1.26
Photons	0.315	0.85	0.98

As detailed in Sec. 3.2.3.3, GPUMCD can take advantage of a system with multiple GPUs. The impact of parallelizing GPUMCD on two GPUs is presented in Tab. 3.6 where TPH_s and TPH_d are the execution time in single and dual GPU mode, respectively, normalized to the time required for the simulation of a single particle. The graphics cards used for this test are a pair of Tesla C1060. The reported execution times include every memory transfer to and from the graphics cards.

Table 3.6 Execution times and acceleration factors achieved with a multi-GPU configuration.

	TPH_s (μs)	TPH_d (μs)	$A_{d/s}$
Electrons	5.43	2.82	1.92
Photons	5.20	2.74	1.90

3.4 Discussion

3.4.1 Dosimetric evaluation

Figures 3.1 - 3.3 and Tables 3.1 - 3.3 present the dosimetric evaluation performed between GPUMCD and EGSnrc. An excellent agreement is observed for homogeneous phantoms with both photon and electron beams, except for the end of the build-up region where small

differences can be observed. In both cases, the rest of the curve as well as the overall range of the particles are not affected.

In heterogeneous simulations with low density materials, the agreement is also excellent. Differences of up to 2% can be seen in the build-up region for an incoming electron beam. These differences again do not affect the range of the particles. For a photon beam impinging on the water-lung phantom, the dose is in good agreement within as well as near the heterogeneity.

In heterogeneous simulations with a high-Z material, differences of up to 2% are found with an impinging electron beam and no differences above 1% are found for the photon beam in the presented PDD while differences of up to 1.5% are found in the profiles.

The overall quality of the dose comparison can be evaluated with the gamma value results presented in Tab. 3.1 to 3.3. Results suggest that the code is suitable for clinical applications as the dose accuracy is within the 2%/2 mm criteria, for all cases but one, which is below the recommended calculation accuracy proposed by Van Dyk (Van Dyk et al., 1999). For electron beams, the gamma criteria has been well respected in the first two geometries where at most 2% of the significant voxels failed. More pronounced differences have been found in the last geometry including a high-Z material where 8% of the significant voxels fail the gamma test. However, only 3% of all significant voxels fail that test with a value higher than 1.2, indicating that the critera are slightly too strict. Indeed, with a 2.5%-2.5mm, 3% of all significant voxels fail the test. For photon beams, all geometries meet the gamma criteria for 98 % or more of all significant voxels.

3.4.2 Execution times

The differences in calculation efficiency between GPUMCD and the other two codes are due to three distinctive reasons: 1) differences in hardware computational power, 2) differences in the programming model and its adaptation to hardware and 3) differences in the physics simulated and approximations made. As an operation-wise equivalent CPU implementation

of GPUMCD does not exist, these factors cannot be evaluated individually.

Tab. 3.4 shows the acceleration results comparing GPUMCD to DPM and EGSnrc. It can be seen that GPUMCD is consistently faster than DPM, by a factor of at least 262x for photon beams and 276x for electron beams. The disadvantage of using the Woodcock raytracing algorithm is apparent in Tab 3.4 where the geometry featuring a heavier than water slab has the lowest acceleration factor. An acceleration of more 1200x is observed when comparing to EGSnrc for both photon and electron beams.

DPM and EGSnrc both employ much more sophisticated electron step algorithms compared to GPUMCD. Improvements to the electron-step algorithm in GPUMCD will most likely yield greater accelerations, as photon transport was shown to be much faster than the reported coupled transport results (Hissoiny et al., 2010b), as well as improved dosimetric results.

Tab. 3.5 presents the execution times and acceleration factors for the different divergence reduction methods and optimizations presented in Sec. 3.2.3.3. Gains of up to 26% for electron beams are observed. However, acceleration factors below 1 are found for photon beams, showing that electron beams are more divergent. These results show that the GPU architecture is not ideal for Monte Carlo simulations but that the hardware is able to cope well with this divergence. The mechanisms employed in GPUMCD to reduce this penalty have either had a negative impact on the execution time or a small positive impact that is perhaps not worth the loss in code readability and maintainability.

Tab. 3.6 details the gain that can be obtained by executing GPUMCD on multiple graphics card. GPUMCD presents a linear growth with respect to the number of GPUs. This is expected as there is a relatively small amount of data transferred to the GPUs compared to the amount of computations required in a typical simulation. As the number of histories is reduced, so is the acceleration factor.

Efficiency improvements can be achieved with variance reduction techniques (VRT). Several methods have been published to improve the calculation efficiency of EGSnrc for instance (Kawrakow and Rogers, 2000; Wulff et al., 2008). A fair comparison of GPUMCD

with other Monte Carlo packages would ideally involve an optimal usage of the codes. In this paper, no VRT was used for the comparisons. It is not clear yet how VRT could potentially be implemented in GPUMCD. In the eventuality that VRT are equally applicable to each code architecture, the efficiency improvements reported in this paper would remain roughly the same. Otherwise, GPUMCD might suffer from a performance penalty compared to VRT-enabled codes.

3.5 Conclusion and future work

In this paper, a new fully coupled GPU-oriented Monte Carlo dose calculation platform was introduced. The accuracy of the code was evaluated with various geometries and compared to EGSnrc, a thoroughly validated Monte Carlo package. The overall speed of the platform was compared to DPM, an established fast Monte Carlo simulation package for dose calculations. For a 2%-2mm gamma criteria, the dose comparison is in agreement for 98% or more of all significant voxels, except in one case: the tissue-bone-lung phantom with an electron beam where 92% of significant voxels pass the gamma criteria. These results suggest that GPUMCD is suitable for clinical use.

The execution speed achieved by GPUMCD, at least two orders of magnitude faster than DPM, let envision the use of accurate of Monte Carlo dose calculations for numerically intense applications such as IMRT or arc therapy optimizations. A 15 MeV electron beam dose calculation in water can be performed in less than 0.12 s for 1M histories while a photon beam calculation takes less than 0.27 s for 4M histories.

GPUMCD is currently under active development. Future work will include the ability to work with phase space and spectrum files as well as the integration and validation of CT-based phantoms for dose calculation. Research towards variance reduction techniques that are compatible with the GPU architecture is also planned.

Acknowledgment

This work was supported by Natural Sciences and Engineering Research Council of Canada (NSERC). Dual GPU computations were performed on computers of the Réseau québécois de calcul de haute performance (RQCHP). Multiple scattering data has been provided by the National Research Council of Canada (NRC).

CHAPITRE 4

VALIDATION OF GPUMCD FOR LOW ENERGY BRACHYTHERAPY SEED DOSIMETRY

Sami Hissoiny

École polytechnique de Montréal, Département de génie informatique et génie logiciel, 2500 chemin de Polytechnique. Montréal (Québec), CANADA H3T 1J4

Benoît Ozell

École polytechnique de Montréal, Département de génie informatique et génie logiciel, 2500 chemin de Polytechnique. Montréal (Québec), CANADA H3T 1J4

Philippe Després

Département de radio-oncologie, Centre hospitalier universitaire de Québec (CHUQ), 11 Côte du Palais, Québec (Québec), Canada G1R 2J6

Jean-François Carrier

Département de Physique, Université de Montréal, Pavillon Roger-Gaudry (D-428), 2900 Boulevard Édouard-Montpetit, Montréal, Québec H3T 1J4, Canada and Département de Radio-oncologie, Centre hospitalier de l'Université de Montréal (CHUM), 1560 Sherbrooke est, Montréal, Québec H2L 4M1, Canada

ABSTRACT

Purpose : To validate GPUMCD, a new package for fast Monte Carlo dose calculations based on the GPU (Graphics Processing Unit), as a tool for low energy single seed brachytherapy dosimetry for specific seed models. As the currently accepted method of dose calculation in low energy brachytherapy computations relies on severe approximations, a Monte Carlo based approach would result in more accurate dose calculations, taking in consideration the patient anatomy as well as inter-seed attenuation. The first step is to evaluate the capability of GPUMCD to reproduce low energy, single source, brachytherapy calculations which could ultimately result in fast and accurate, Monte Carlo based, brachytherapy dose calculations for routine planning.

Method : A mixed geometry engine was integrated to GPUMCD capable of handling parametric as well as voxelized geometries. In order to evaluate GPUMCD for brachytherapy calculations, several dosimetry parameters were computed and compared to values found in the literature. These parameters, defined by the AAPM Task-Group No. 43, are the radial dose function, the 2D anisotropy function, and the dose rate constant. These three parameters were computed for two different brachytherapy sources: the Amersham OncoSeed 6711 and the Imagyn IsoStar IS-12501.

Results : GPUMCD was shown to yield dosimetric parameters similar to those found in the literature. It reproduces radial dose functions to within 1.25% for both sources in the $0.5 < r < 10$ cm range. The 2D anisotropy function was found to be within 3% at $r = 5$ cm and within 4% at $r = 1\ cm$. The dose rate constants obtained were within the range of other values reported in the literature.

Conclusion : GPUMCD was shown to be able to reproduce various TG-43 parameters for two different low-energy brachytherapy sources found in the literature. The next step is to test GPUMCD as a fast clinical Monte Carlo brachytherapy dose calculations with multiple seeds and patient geometry, potentially providing more accurate results than the TG-43 formalism while being much faster than calculations using general purpose Monte Carlo codes.

4.1 Introduction

In permanent implant brachytherapy, a number of radioactive seeds are implanted directly into the structure targeted for treatment. In clinical practice, the formalism currently employed to compute the dose in brachytherapy cases is outlined in the AAPM Task Group No. 43 (TG-43) Report (Rivard et al., 2004). The technique relies on Monte Carlo parameters that are pre-calculated in a homogeneous water geometry used to approximate the patient.

The shortcomings of this technique are evident : the patient is not made of water nor is he homogeneous. One of the alternatives to the TG-43 formalism is to perform full Monte Carlo simulations based on the patient's anatomy. For low dose rate prostate brachytherapy, differences between full Monte Carlo calculations and TG-43 calculations arise from two main contributions: inter-seed attenuation and patient composition (Carrier et al., 2006). The inter-seed attenuation stems from the fact that several seeds made of high-Z materials are implanted in the patient, shadowing each other and leading to fluence perturbations ignored in the TG-43 formalism. The patient composition differences come from the fact that the patient is not homogeneously composed of water, as approximated in the TG-43 formalism.

Several studies have shown that the differences between full Monte Carlo and TG-43 are significant (Carrier et al., 2006; Mobit and Badragan, 2004; Chibani et al., 2005). Carrier *et al.* reported that the difference to the prostate's D_{90} can range from 5.8% to 12.8% (Carrier et al., 2006). Mobit *et al.* have shown, in a non clinically realistic setting, that the dose perturbation due to inter-seed attenuation in a 27 seeds plan can be as high as 10%. The effect was not significant with seed spacing of more than 0.5 cm but significant differences were found in the dose-volume histogram for seeds spacing of 0.5 cm(Mobit and Badragan, 2004). Chibani *et al.* have compared full Monte Carlo simulations to a kernel-based superposition approach using idealized point or line sources (Chibani et al., 2005). They have found differences in the D_{90} of 2% to 5%.

Despite these differences, full Monte Carlo simulations are rarely used routinely due to their typically long computation times. Yegin *et al.* have designed a faster Monte Carlo code

based on EGSnrc (Kawrakow and Rogers, 2000) called Brachydose. It was reported that the platform could complete a calculation in 30 secs for prostate implants (Thomson et al., 2010). This is still much longer than the time required for a regular TG-43 calculation.

GPUMCD is a fast Monte Carlo code relying on the GPU (Graphics Processing Unit) for all arithmetic operations (Hissoiny et al., 2011a). It was shown to produce clinically acceptable results, comparable to those of EGSnrc for both photon and electron beams but obtained up to 1500 times faster.

A mixed geometry engine was integrated to GPUMCD, capable of handling parametric as well as voxelized geometries. This made GPUMCD suitable for brachytherapy computations, for which seeds are best described with parametric geometries while patient specific data usually take the form of voxelized data sets obtained by imaging.

The main objective of this paper is to evaluate the capacity of GPUMCD to reproduce dosimetry results of single low energy brachytherapy seeds. The TG-43 requires for validation that any new dose calculation tool should be able to reproduce the currently accepted TG-43 parameters for at least one source. In this report, two sources were validated: the Amersham OncoSeed 6711 and the Imagyn IsoStar IS-12501.

The goal of this paper is not to introduce a new set of TG-43 parameters for the presented sources. The low- energy physics modeling of GPUMCD is not as complete, or as accurate, as other sources of low-energy brachytherapy Monte Carlo simulations such a MCTP (Williamson and Quintero, 1988) or EGSnrc (Kawrakow and Rogers, 2000). This study will however serve as a basis to evaluate the accuracy that can be reached by using GPUMCD as clinical low-energy brachytherapy dose calculation tool. The areas where the low-energy physics in GPUMCD falls short of what would be preferable in a brachytherapy context will be highlighted in Sec 5.2.

This study is not focused on execution times but rather on the evaluation of GPUMCD as a tool for producing accurate results in brachytherapy simulations. It is the first step towards the integration of GPUMCD as a clinically useful tool. A subsequent evaluation, outside the

scope of this study, will assess the ability of GPUMCD to complete Monte Carlo simulations of brachytherapy problems in a time compatible with routine clinical use.

The rest of this paper is structured as follows: Sec. 4.2.1 reviews the important aspects of the GPUMCD platform for brachytherapy, Sec. 4.2.2 describes the brachytherapy elements added to GPUMCD since its external beam version (Hissoiny et al., 2011a) and Sec. 4.2.3 details the brachytherapy seeds used in this study. Sec. 5.3 presents the results obtained for the two sources and the different TG-43 parameters as well as a discussion of those results.

4.2 Material and methods

4.2.1 Review of GPUMCD features relevant to brachytherapy computations

The cross section data for this version of GPUMCD comes from the NIST XCOM database (Berger et al., 1998), as recommended by the TG-43 report.

Rayleigh scattering is not currently part of the GPUMCD platform as it was originally designed for high-energy external beam radiotherapy. Rayleigh scattering is less present than either Compton scattering or the photoelectric effect throughout the energy range. In high-Z materials, such as the ones found in the seeds, Rayleigh scattering is dominated by the photoelectric effect by close to two orders of magnitude. It is however recommended by the TG-43 report that Rayleigh scattering be part of single-seed brachytherapy dosimetry Monte Carlo calculations.

The Compton scattering event was modeled using a free electron approximation through the sampling of the Klein-Nishina cross section (Klein and Nishina, 1929). The direction and energy of the scattered photon were sampled using the algorithm by Everett *et al.* (Everett et al., 1971). The ejected electron is deposited locally. At the energies encountered in low-energy brachytherapy, it is the recommendation of the TG-43 report that a Compton scattering algorithm taking into consideration bound electrons is preferable.

The photoelectric effect was modeled by ignoring all atomic relaxations and characteristic radiation in the original GPUMCD code. These omissions had a minor impact in external beam radiotherapy but are important at the energies considered in permanent seed implant brachytherapy. A minor modification was made to the original GPUMCD and is detailed in Sec. 4.2.2.

All the energy of the secondary electrons is deposited locally in this study since their range would be much smaller than the resolution of the scoring grid used. This approximation is valid due to the low energy of the primary photons. At 35 keV of kinetic energy, the range of an electron is 2.3x10^{-3} g/cm^2 in water (Berger et al., 2000), which is much less than the voxel size in the scoring grid. Photons are tracked until they reach their cutoff value of 1 keV, after which their energy is deposited locally.

To handle heterogeneities, GPUMCD uses the Woodcock raytracing algorithm (Carter et al., 1972; Hissoiny et al., 2011a) in which the volume can be considered homogeneously attenuating with a fictitious attenuation coefficient, corresponding to a fictitious interaction which leaves the direction and energy of the particle unchanged, in every voxel $\vec{x}$:

$$\mu(\vec{x}, E)_{fict} = \mu(E)_{max} - \mu(\vec{x}, E), \tag{4.1}$$

where μ_{max} is the attenuation coefficient of the most attenuating region inside the treatment volume. The distance to the next interaction is therefore always sampled using μ_{max}.

This raises a problem in permanent implant brachytherapy calculations since there is a very dense but very localized material in the simulation: the seed. This small region of space will be responsible for the slowing down of the entire simulation due to the very high number of fictitious interactions introduced everywhere else since $\mu(E)_{max}$ is much larger than $\overline{\mu(\vec{x}, E)}$, the average attenuation factor of the simulation volume. To address this issue, a completely new boundary crossing algorithm was introduced in GPUMCD. It is presented in Sec. 4.2.2.2.

4.2.2 Additions to the GPUMCD platform

The first modification, presented in Sec. 4.2.2.1, is the addition of a parametric geometry module. The second one, presented in Sec. 4.2.2.2, is the new boundary crossing algorithm. The third one, presented in Sec. 4.2.2.3, is the modification of the photoelectric effect simulation to handle characteristic radiation.

4.2.2.1 Geometry module

Permanent implant brachytherapy seeds are unsuitable for a representation using a voxelized grid due to their small size and curved surfaces.

A new parametric geometry module based on quadrics has been developed. Because the quadrics can be infinite in space (*e.g.* a cylinder) the possibility of a parametric box is also added to the geometry package. This box is used to bind otherwise infinite regions of space.

The parametric geometry package can be used jointly with the regular voxelized grid definition of GPUMCD. This allows the use of voxel-based CT data defining patient composition and density with parametric seed geometry definitions.

The navigation in the mixed geometry required a modification to the photon simulation algorithm. During the simulation, the algorithm needs to retrieve the composition and the density at the current particle position. To handle the mixed geometry, GPUMCD first does a test against every parametric volume to verify if the current particle position is within one of them. If it is not the case, then the geometry engine assumes that the composition and the density are specified by the underlying volume of voxels and fetches these values from the phantom 3D textures (for more details see Sec. II.C.2 of (Hissoiny et al., 2011a)).

This approach raises two noteworthy points. Firstly, the parametric definitions have to be defined in a specific order, from inside out. That is, if the volume V_1 is surrounded by volume V_2, then V_1 must be defined before V_2. Secondly, the systematic search for a position within

the parametric geometries is time consuming, especially since typical treatment plans involve tens of seeds each composed of many parametric volumes.

The parametric definitions were stored in the constant memory of the GPU. Since these definitions will always be read in the same order and that every thread will access the same element at the same time, the bandwidth-saving broadcasting mechanism of constant memory can be used. However, this requires the number of parametric definitions to be known at the time of compilation since constant memory has to be statically allocated.

Also, to limit the performance penalty associated with the systematic search through parametric volumes, a bounding box was placed around each and every seed. The geometry engine first checks if the particle is within the bounding box and will only verify the different seed quadrics if this is the case.

4.2.2.2 Boundary crossing algorithm

In the version of `GPUMCD` for brachytherapy calculations, the new quadric geometry engine is used for the heterogeneous boundary crossing during the simulation. The approach used in `GPUMCD` involves the rescaling of a sampled number of mean free paths to the next interaction through the conservation of the non-interaction probability (Leppanen, 2007):

$$e^{-x_i\Sigma_i} = e^{-x_j\Sigma_j}, \tag{4.2}$$

where x_i and Σ_i are distances traveled and total cross sections for different media, respectively. From this relation, the distances in medium i can be rescaled to be equivalent, in terms of interaction probability, to a distance traveled in medium j through the following relation:

$$x_j = \frac{\Sigma_i x_i}{\Sigma_j}. \tag{4.3}$$

With these equations, if a path length s is sampled in a medium of total cross section Σ_0, the actual distance traveled must respect the following relation:

$$s = \sum_j \frac{\Sigma_j x_j}{\Sigma_0}, \tag{4.4}$$

where the j's represent the media crossed. The actual traveled distance (in units of length) to the next point of interaction, x, is then

$$x = \sum_j x_j. \tag{4.5}$$

The approach to compute the x_j's is a raytracing problem. The mathematical tools required for the computation of the x_j's are a ray-box intersection algorithm (Woo, 1990) and a ray-quadric intersection algorithm (Lindley, 1992).

The Woodcock algorithm is however still used when the particle is outside a quadric until the particle reaches the next quadric.

4.2.2.3 Modification to the photoelectric effect simulation

In the original `GPUMCD` code (Hissoiny et al., 2011a), the photoelectric effect was approximated by E_e=E_p+m_0, where m_0 is the rest mass of the electron. The energy of the incoming photon, E_p, was completely transferred as the energy of the ejected electron, E_e. This approximation is acceptable with high energy beams but as the energy of primary particles approaches the energy of the characteristic X-ray radiation of the material, it becomes less accurate. In this study, the direction of the ejected electron is irrelevant since it is not tracked. The required changes are related to the energy of the electron absorbed locally and the energy of the relaxation X-ray photon.

In the version of `GPUMCD` for brachytherapy, the energy deposited locally (since electrons are not tracked) is set to $E_d = E_p - E_{k-edge}$, where E_d is the energy deposited and E_{k-edge} is the binding energy of the K-shell. This approach is also used in EGS4 (Nelson et al.,

1985). The incoming photon is recast as a relaxation X-ray photon; its energy is set to $E_p = E_{k-edge}$ and its direction is sampled uniformly. With this approach, the relaxation photon is simulated right away and no new photon is added to the particle queue, therefore avoiding a performance penalty. The photoelectric algorithm also takes into account K-shell yields to sample if a fluorescence X-ray should be generated based on Plechaty *et al.* (Plechaty et al., 1978).

As a single E_{k-edge} value is expected by the algorithm, it cannot readily handle compounds where each element of the compound has its own E_{k-edge}. A more complex algorithm would require the sampling of the element from which the electron is ejected, taking into consideration the photoelectric cross section of each element, as it is done in EGSnrc (Kawrakow and Rogers, 2003), for instance.

4.2.3 Brachytherapy seed description

Two brachytherapy sources were used in this study: the Amersham OncoSeed 6711 and the Imagyn IsoStar IS-12501. Both of these seeds have ^{125}I as their radioactive source of photons. The five bin energy spectrum presented in the TG-43 report (Rivard et al., 2004) was used in this study.

The detailed description of the OncoSeed 6711 source is taken from the study by Dolan *et al.* (Dolan et al., 2006). The source consists AgI and AgBr coated on a 2.88 mm long cylindrical rod of silver ($\rho = 10.5$ g/cm^3). The rod has a radius of 0.25 mm. The ends of the rod are cut at a 45 degrees angle yielding a radius of 0.175 mm at both ends of the rod. The rod is encapsulated inside a titanium ($\rho = 4.54$ g/cm^3) shell with a radius of 0.4 mm. The wall of the shell is 0.07 mm thick and the capsule is 3.75 mm long. Semi-spherical end-welds with a radius of 0.4 mm and a thickness of 0.375 mm complete the description of the seed.

The detailed description of the IsoStar IS-12501 source is available in the study by Gearheart *et al.* (Gearheart et al., 2000). The source consists of 5 silver spheres coated with AgI. The spheres have a radius of 0.32 mm. The spheres are encapsulated in a titanium shell

that is 3.60 mm long, has a 0.4 mm radius and 0.05 mm thick walls. The end-welds are hemispheres with a radius of 0.4 mm.

A source of uncertainty in Monte Carlo simulations of brachytherapy seeds is the geometric uncertainty, or component mobility, which models the fact that components within seed are not always precisely at their nominal position. Studies by Williamson and Rivard (Williamson, 2002; Rivard, 2001) demonstrate the effect of such uncertainties on the Monte Carlo results. The geometric uncertainty of the seed is not taken into account in this study and therefore every seed is modeled at its nominal geometric configuration. The same goes for the comparison benchmark data of Sec 5.3.

4.2.4 Performance evaluation

4.2.4.1 Monte Carlo simulations

The values of $D(x,y,z)\cdot r^2$, where $D(x,y,z)$ is the energy deposited by a photon at location (x,y,z) at distance r from the center of the source to (x,y,z), are scored directly within the simulation. This approach is used since most TG-43 parameters require a normalization by $G_P(r,\theta)$.

The phantom used for the computation of $g_P(r)$, $F(r,\theta)$ at $r = 1$ cm and $r = 5$ cm is a 30x30x30 cm^3 box of water ($\rho = 0.998$ g/cm^3), complying with the 15 cm radius water sphere recommended in the TG-43.

The energy deposition was scored using the inherent cylindrical symmetry of the sources. The energy deposition on an annulus of radius r and thickness equal to a voxel side, centered on the source's central axis, is collapsed to a plane going through the source's central axis. The underlying dose grid is then effectively collapsed to a 1-slice Cartesian grid of 257x257 voxels. This number yields for voxel sizes a little over 1 mm^3. The energy deposition estimator used is an analogue estimator. For every simulation, $100\cdot 10^6$ particles were simulated.

Calculations for the air-kerma rate $\dot{K}_\delta(d)$ are done *in vacuo*. A scoring geometry similar to the NIST WAFAC (Wide-Angle Free-Air Chamber) (Seltzer et al., 2003) has been employed, with a 2.7x2.7x0.05 cm^3 voxel positioned at 10 cm from the source on the transverse axis. The voxel is filled with dry air while the rest of the geometry is filled with void. This method is based on the WAFAC method described by Taylor *et al.* (Taylor and Rogers, 2008). The TG-43 report also recommends to set the photon cutoff value to 5 keV to eliminate relaxation photons created in titanium shells. In GPUMCD, the atomic relaxation in titanium can be turned off independently while still allowing the tracking of lower energy photons that have not been generated in the titanium shell. A lower cutoff value of 1 keV was chosen, allowing for a more accurate simulation while respecting the TG-43U recommendation of not tracking titanium shell relaxation photons. For the calculation of the dose rate constant Λ, the dose rate at the reference position $\dot{D}(r_0, \theta_0)$ is divided by the air-kerma strength, $S_k = \dot{K}_\delta(d)d^2$, at $d = 10$ cm.

4.2.4.2 Simulation setup and validation

The simulations were run on a PC equipped with a NVIDIA GeForce GTX 480 GPU and an Intel E6550 CPU. The comparison data for validation were taken from the point approximated results of the "CLRP TG-43 Parameter Database" (Taylor and Rogers, 2008). The values in this database were obtained with BrachyDose (Taylor et al., 2007). The TG-43 report also presents consensus data for the two types of sources used and these values will also be compared to results obtained in this study. The agreement evaluation in the following section always refer to the difference between `GPUMCD` and BrachyDose.

4.3 Results and Discussion

4.3.1 OncoSeed 6711

The dose rate constant result is presented in Tab. 4.1. The reported value is within 1% of the WAFAC value calculated by Taylor*et al.* (Taylor and Rogers, 2008).

The statistical uncertainty of the Monte Carlo simulation, $\sigma_{\Lambda|s}$, is 0.4%. This statistical uncertainty is one of the factors used to estimate the overall uncertainty of Λ (See Sec. IV.C of (Rivard et al., 2004)). The other factors, $\sigma_{\Lambda|\mu}$ relating to the cross-section uncertainty and $\sigma_{\Lambda|geo}$ representing the geometrical uncertainty, have not been evaluated.

Table 4.1 Dose rate constant value for the OncoSeed 6711 source.

Authors	Method	Value ($cGyh^{-1}U^{-1}$)	$\sigma_{\Lambda\|s}$
Taylor *et al.* (Taylor et al., 2007)	WAFAC	0.924	0.002
Taylor *et al.* (Taylor et al., 2007)	point	0.942	0.003
Dolan *et al.* (Dolan et al., 2006)	extrap (PTRAN)	0.942	0.017
Dolan *et al.* (Dolan et al., 2006)	TLD	0.971	0.059
Rivard *et al.* (Rivard et al., 2004)	TG-43 Consensus Value	0.965	
this work	**GPUMCD, WAFAC**	**0.927**	**0.004**

In Fig. 4.1, the results for the computation of the $g_P(r)$ function are presented. Excluding the region of very high gradient ($0.05 < r < 0.5$ cm) where the voxel size is too coarse to draw any conclusion, the difference between the results computed in this study and the results presented by Taylor *et al.* is at most 1.25%. The statistical uncertainty is 0.5% at $r = 1$ cm and 1.2% at $r = 10$ cm.

Two anisotropy functions $F(r, \theta)$ are presented in Fig 4.2 at distances $r = 1$ cm and $r = 5$ cm.

At $r = 1$ cm, an agreement within 4% is observed throughout the θ range. In the range $(0 < \theta < 0.5)$, the results from this study are within the results obtained with BrachyDose and those of the TG-43 consensus dataset. In the range $(0.5 < \theta < \pi/2)$, the results obtained are constantly below those obtained by BrachyDose and the TG-43 dataset, by less than 4%.

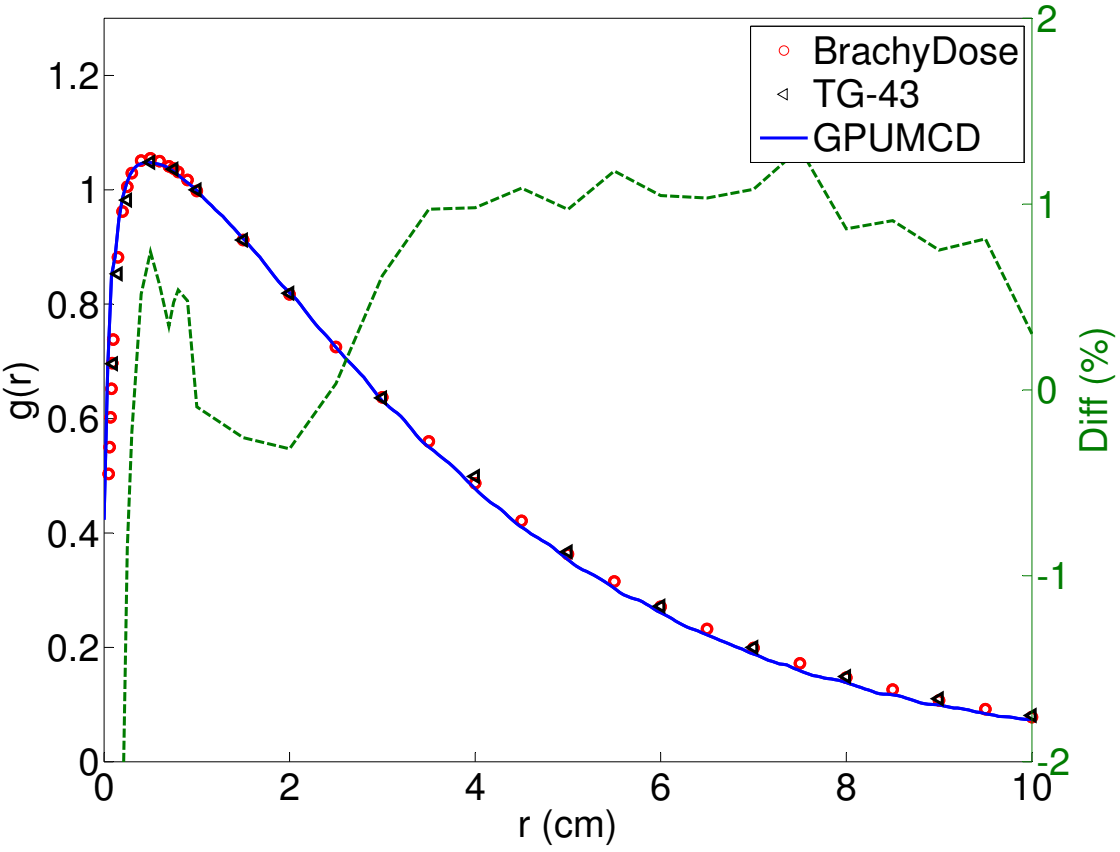

Figure 4.1 The $g_P(r)$ function for the OncoSeed 6711 source. Dashed line gives % difference between the values computed within this study and those of BrachyDose.

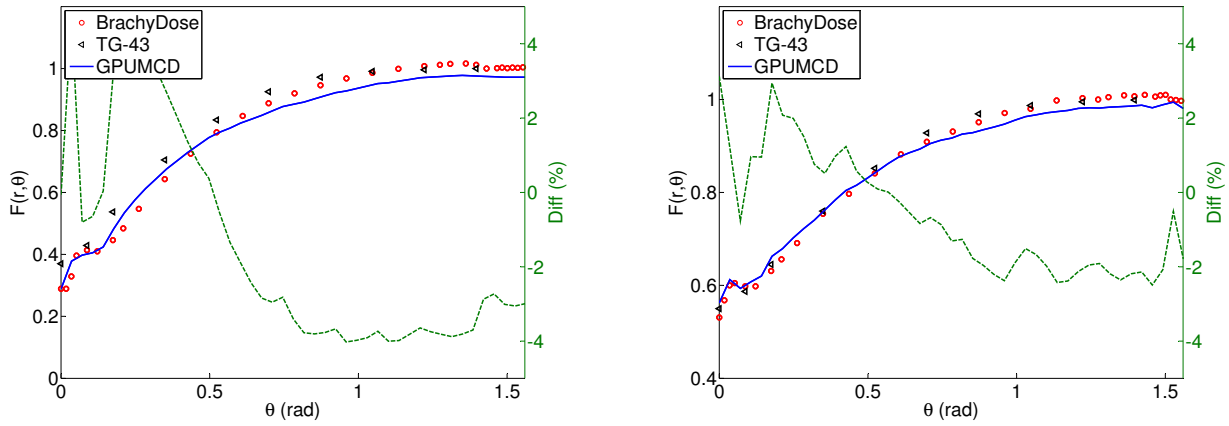

Figure 4.2 The $F(r,\theta)$ for the OncoSeed 6711 source at $r = 1$ cm (left) and $r = 5$ cm (right). Dashed line gives % difference between the values computed within this study and those of BrachyDose.

The statistical uncertainties at $\theta = 0$ and $\theta = \pi/2$ are 0.7% and 0.6% respectively.

At $r = 5$ cm, differences of less than 3% were observed throughout the angular range. For a majority of the angular range ($0.3 < \theta < \pi/2$), the differences are below 2.3%. The statistical uncertainties at $\theta = 0$ and $\theta = \pi/2$ are 1.1% and 0.6% respectively.

4.3.2 Imagyn IsoStar IS-12501

The dose rate constant result is presented in Tab. 4.2, along with data from the Brachytherapy TG-43 database (Taylor and Rogers, 2008). The statistical uncertainty of the simulation is 0.5%.

In Fig. 4.3, the results for the computation of the $g_P(r)$ function are presented. The difference between the results computed for this study and the results presented by Taylor *et al.* is at most 1.25%, excluding the region of very high gradient near $r = 0$ cm where the voxel sizes are too large to draw conclusions. The statistical uncertainties at $r = 1$ cm is 0.3% and at $r = 10$ cm is 0.9%.

Two anisotropy functions $F(r,\theta)$ are presented in Fig 4.4 at distances $r = 1$ cm and $r = 5$ cm.

Table 4.2 Dose rate constant values for the IsoStar IS-12501 source.

Authors	Method	Value ($cGyh^{-1}U^{-1}$)	$\sigma_{\Lambda\|s}$
Taylor *et al.* (Taylor et al., 2007)	WAFAC	0.924	0.003
Taylor *et al.* (Taylor et al., 2007)	point	0.923	0.003
Ibbott *et al.* (Ibbott et al., 2002)	point	0.92	-
Ibbott *et al.* (Ibbott and Nath, 2001)	TLD	0.91	0.07
Nath *et al.* (Nath and Yue, 2000)	TLD	0.95	0.095
Rivard *et al.* (Rivard et al., 2004)	TG-43 Consensus Value	0.940	-
this work	**GPUMCD, WAFAC**	**0.926**	**0.005**

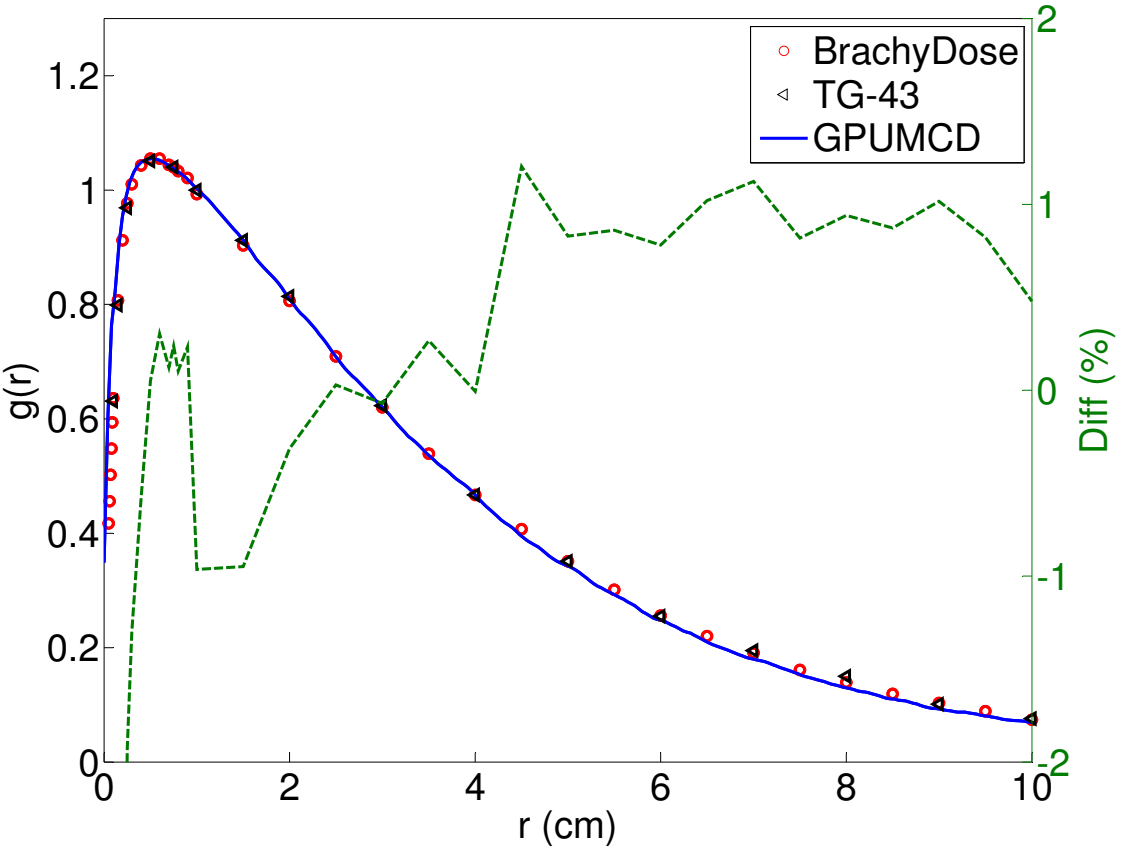

Figure 4.3 The $g_P(r)$ function for the IsoStar IS-12501 source. Dashed line gives % difference between the values computed within this study and those of BrachyDose.

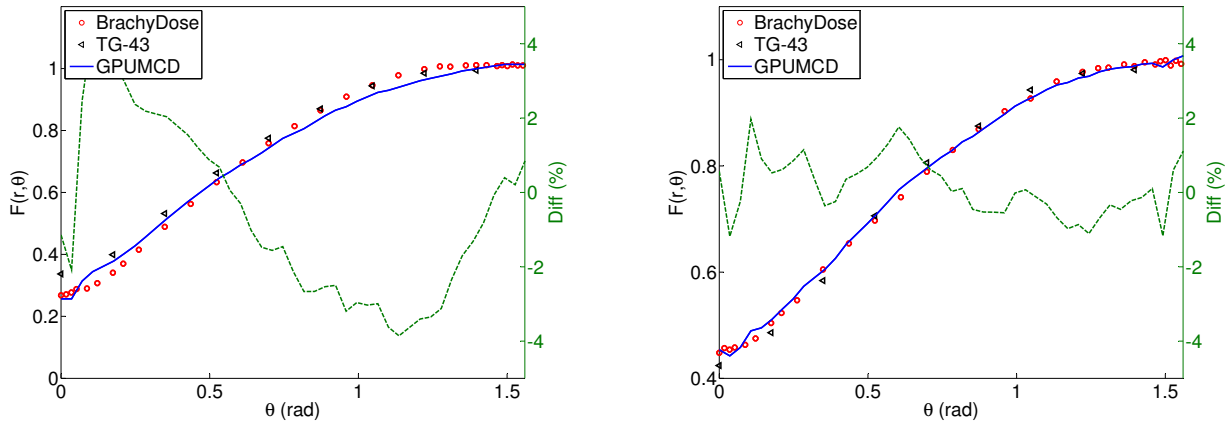

Figure 4.4 The $F(r,\theta)$ for the IsoStar IS-12501 source at $r = 1$ cm (left) and $r = 5$ cm (right). Dashed line gives % difference between the values computed within this study and those of BrachyDose.

At $r = 1$ cm, in the regions covered by $0.25 < \theta < \pi/2$ an agreement within 4% is observed. The difference is again larger in the endcap region for $0 < \theta < 0.25$ where local differences of up to 4.2% can be found. The statistical uncertainties at $\theta = 0$ and $\theta = \pi/2$ are 0.8% and 0.5% respectively.

At $r = 5$ cm, differences of less than 2% are found in the range $0 < \theta < \pi/2$. The general shape of the distribution around the end cap is similar when comparing both algorithms. The statistical uncertainties at $\theta = 0$ and $\theta = \pi/2$ are 1.2% and 0.8% respectively.

4.3.3 Execution times

This study does not focus on execution times and therefore a comparison to established Monte Carlo codes was not thoroughly made. However, the main purpose of the GPU as a computing platform is the gain in execution time. For completeness, the execution times have been collected. The time to simulate 1 M histories for the OncoSeed 6711 is 61 ms while it is 85 ms for the IsoStar IS-12501. The difference can be explained by the added number of geometrical definitions needed to model the IsoStar IS-12501.

4.4 Conclusion and future work

GPUMCD is a new GPU-based Monte Carlo platform originally developed for external beam dose calculations. In this study, GPUMCD was shown to be suitable for low energy brachytherapy single seed dosimetry using two different brachytherapy sources : an OncoSeed 6711 and a Imagyn IsoStar IS-12501. Three different TG-43 parameters have been computed: the radial dose function, the 2D anisotropy function and the dose rate constant. Throughout these tests, GPUMCD has been shown to replicate radial dose functions within 1.25% in the $0.5 < r < 10$ cm range and 2D anisotropy functions, for most of the angular range, within 4% at $r = 1$ cm and within 3% at $r = 5$ cm when compared to values accepted in the literature.

GPUMCD has been able to provide accurate results for these two seeds, which have relatively different geometries. It is a good indication that it is capable of delivering accurate results for any seed geometry.

These results however cannot be extrapolated to any seed type, with any material, or any radionuclide. The approximations used in the photoelectric process as well at the fact that bound electrons and Rayleigh scattering are not taken into consideration could potentially break down for sources with different materials and a different energy spectrum. The exercise described in this study should therefore be repeated for every new type of seed used with GPUMCD.

This study is a first step in studying the overall usability of GPUMCD for routine clinical brachytherapy computations. In a future study, complete plans in patient geometries will be computed using GPUMCD and the execution time required to complete the calculation will be studied.

Acknowledgments

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC). Graphics hardware was kindly donated by NVIDIA Corporation.

CHAPITRE 5

FAST DOSE CALCULATION IN MAGNETIC FIELDS WITH GPUMCD

Sami Hissoiny

École polytechnique de Montréal, Département de génie informatique et génie logiciel, 2500 chemin de Polytechnique. Montréal (Québec), CANADA H3T 1J4

Alexander Raaijmakers

Department of Radiotherapy, University Medical Center Utrecht, Heidelberglaan 100, 3584 CX, Utrecht, The Netherlands

Benoît Ozell

École polytechnique de Montréal, Département de génie informatique et génie logiciel, 2500 chemin de Polytechnique. Montréal (Québec), CANADA H3T 1J4

Philippe Després

Département de radio-oncologie, Centre hospitalier universitaire de Québec (CHUQ), 11 Côte du Palais, Québec (Québec), Canada G1R 2J6

Bas Raaymakers

Department of Radiotherapy, University Medical Center Utrecht, Heidelberglaan 100, 3584 CX, Utrecht, The Netherlands

ABSTRACT

A new hybrid imaging-treatment modality, the MRI-Linac, involves the irradiation of the patient in the presence of a strong magnetic field. This field acts on the charged particles, responsible for depositing dose, through the Lorentz force. These conditions require a dose calculation engine capable of taking into consideration the effect of the magnetic field on the dose distribution during the planning stage. Also, in the case of a change in anatomy at the time of treatment, a fast online replanning tool is desirable. It is improbable that analytical solutions such as pencil beam calculations can be efficiently adapted for dose calculations within a magnetic field. Monte Carlo simulations have therefore been used for the computations but the calculation speed is generally too slow to allow online replanning. In this work, `GPUMCD`, a fast GPU-based Monte Carlo dose calculation platform, was benchmarked with a new feature that allows dose calculations within a magnetic field. As a proof of concept, this new feature is validated against experimental measurements. `GPUMCD` was found to accurately reproduce experimental dose distributions according to a 2%-2mm gamma analysis in two cases with large magnetic field induced dose effects: a depth-dose phantom with an air cavity and a lateral-dose phantom surrounded by air. Furthermore, execution times of less than 15 seconds were achieved for one beam in a prostate case phantom for a 2% statistical uncertainty while less than 20 seconds were required for a 7 beams plan. These results indicate that `GPUMCD` is an interesting candidate, being fast and accurate, for dose calculations for the hybrid MRI-Linac modality.

5.1 Introduction

The UMC Utrecht, in collaboration with Elekta and Philips Medical Systems, is developing a 6 MV linear accelerator system integrated with a 1.5 T MRI (magnetic resonance imaging) scanner (Raaymakers et al., 2009). A similar device, using a 0.2 T magnet, is also being developed at the Cross Cancer Institute in Edmonton (Fallone et al., 2009). A commercial solution is also currently developed by ViewRay (Ohio, USA) using three ^{60}Co units and a

0.35 T magnet.

One of the characteristics of this type of hybrid modality is that the dose delivered by photon beams is affected by the presence of a static and homogeneous magnetic field produced by the MRI. Electrons within the magnetic field have their trajectory affected by the presence of the magnetic field (Raaijmakers et al., 2005).

The aim of such systems is to allow real-time position verification with superior soft-tissue contrast (Lagendijk et al., 2008). The dose that had originally been planned may have to be recomputed due to a change in the anatomy of the patient, also detectable through the use of online MRI imaging. A dose calculation engine must therefore be able to take into consideration the presence of the permanent magnetic field but also deliver results in a timeframe compatible with an online replanning approach if the plan has to be recomputed due to changes detected by the imaging modality. MRI acquired images can be segmented and used as the input to the dose calculation (Lee et al., 2003).

At this time, Monte Carlo simulations seem to be the only practical method to compute dose distributions in magnetic fields. EGSnrc (Kawrakow and Rogers, 2003), PENELOPE (Salvat et al., 2006) and GEANT4 (Agostinelli et al., 2008) have all been used to compute dose distributions in the presence of magnetic fields (Kirkby et al., 2008; Raaijmakers et al., 2008). GEANT4 and PENELOPE have also been validated through experimental measurements (Raaijmakers et al., 2007; Chen et al., 2005). However, these studies did not focus on the execution times required to compute the results but it is understood that such general purpose Monte Carlo packages are too slow for online replanning.

`GPUMCD` (Hissoiny et al., 2011a) is a new Monte Carlo dose calculation platform which delivers fast and accurate results. It has been designed and programmed from the ground-up to run on GPUs (Graphics Processing Units). It has been compared to EGSnrc and shown to give accurate results for megavoltage beams such as those employed in the MRI-Linac at UMC Utrecht. It has also been shown to be three orders of magnitude faster than EGSnrc.

The current work focuses on the introduction of magnetic field handling in `GPUMCD`, its

validation for dosimetric results as well as the study of its capacity to stay within clinical time constraints. The magnetic field formulation and its incorporation in the electron transport algorithm is presented in section 5.2.1. A comparison to experimentally acquired data for two cases with magnetic field induced dose effects is presented in section 5.2.2. A timing result for a 1-beam and 7-beams prostate phantom dose calculation is performed in section 5.2.3. The objective of this work is to show that `GPUMCD` is accurate and fast enough to be used as the online replanning dose calculation engine for the MRI-Linac project.

5.2 Material and methods

5.2.1 Magnetic field handling in `GPUMCD`

The magnetic field forces on charged particles are described by the Lorentz force equation (Salvat et al., 2006):

$$\vec{F} = \frac{d\vec{p}}{dt} = q(\vec{E} + \frac{\vec{v}}{c} \times \vec{B}), \tag{5.1}$$

where $\vec{E}$ is the electric field and $\vec{B}$ the magnetic field, $\vec{v}$ is the speed of the particle and q its charge. The specific context of Monte Carlo simulations in an MRI-Linac environment can be used to simplify this equation. Firstly, positron tracking is ignored in this study, thus q is always equal to $-e$. Secondly, the MRI requires a homogeneous magnetic field and produces no electric field, it then follows that ($\vec{E} = 0$). Equation 5.1 can be reformulated to:

$$\frac{md(\gamma\vec{v})}{dt} = e(\frac{\vec{v}}{c} \times \vec{B}), \tag{5.2}$$

where $\vec{p} = \gamma m\vec{v}$ and $\gamma = \frac{1}{\sqrt{1-\frac{v^2}{c^2}}}$. The left hand term can be expressed as

$$\frac{md\gamma\vec{v}}{dt} = \frac{\gamma^3 mv}{c^2}\frac{dv}{dt}\vec{v} + \gamma m\frac{d\vec{v}}{dt}, \tag{5.3}$$

where $v = |\vec{v}|$. By using the relation found in equation 5.3 and replacing in equation 5.2 we can find

$$\frac{e}{mc}(\beta\hat{v} \times \vec{B}) = \gamma\beta\frac{d\hat{v}}{dt} + \gamma^3\frac{d\beta}{dt}\hat{v}, \tag{5.4}$$

where $\hat{v} = \vec{v}/|\vec{v}|$ and $\beta = v/c$. The vectors $\hat{v}$ et $\frac{d\hat{v}}{dt}$ are orthogonal and it is therefore possible to decompose equation 5.4 by projecting on these two vectors. Through the first projection we find $\frac{d\beta}{dt} = 0$, which shows that the magnetic field does no work; and through the second projection we find:

$$\frac{d\hat{v}}{dt} = \frac{e}{mc\gamma}(\hat{v} \times \vec{B}). \tag{5.5}$$

The acceleration of the particle is found through

$$a = \frac{d\vec{v}}{dt} = \frac{d(c\hat{v}\beta)}{dt} = c\left(\frac{d\beta}{dt}\hat{v} + \beta\frac{d\hat{v}}{dt}\right) = \frac{e\beta}{m\gamma}(\hat{v} \times \vec{B}), \tag{5.6}$$

and with the first order approximations $\Delta\vec{v} = a_0 t$ and $t = s/v_0$, we find

$$\Delta\vec{v} = \frac{s}{v_0}a_0 = s\frac{e(\hat{v}_0 \times \vec{B})}{m\gamma_0 c}, \tag{5.7}$$

where s is the distance traveled by the particle. Equation 5.7 is expressed in terms that can be integrated into a Monte Carlo simulation.

`GPUMCD` uses Bielajew's random hinge method (Salvat et al., 2006) that can be described as follows: the electron is first transported by ζe_s (where ζ is a uniformly distributed random number), at which point the electron is rotated by the sampled multiple-scattering angle and the energy loss is deposited. The electron is then transported for the remaining distance equal to $(\zeta - 1)e_s$. The magnetic field handling is implemented in `GPUMCD` by evaluating equation 5.7 twice, with $s_1 = \zeta e_s$ and $s_2 = (\zeta - 1)e_s$, during the electron step, before and

after the hinge. The e_s factor is determined by the minimum of the user-defined electron step length limit and the distance to the nearest voxel or parametric geometry definition.

Because the magnetic field is homogeneous, a single three-element vector in constant memory of the GPU is sufficient to store the field information.

5.2.2 Comparison to experimental data

In this study, the results obtained using `GPUMCD` are compared to the experimental results collected by (Raaijmakers et al., 2007).

The validation consisted in using GafChromic film in lateral-dose and depth-dose configurations. The depth-dose configuration used a PMMA-air-PMMA phantom of 8x15x4 cm^3 while the lateral-dose test used a homogeneous PMMA phantom of 6x8x4 cm^3 surrounded by air. The phantoms, as well as an overview of the setup, are detailed in figure 5.1. The phantoms were positioned between the pole shoes of a Bruker 1.3 T magnet. This magnet was installed next to an Elekta SL1 linear accelerator.

Results for magnetic field values of 0 T, 0.6 T and 1.3 T were collected in a previous study by (Raaijmakers et al., 2007) for both phantoms, showing that GEANT4 was producing accurate results when compared to the experimental data using a gamma criterion of 2%-2mm. The results were obtained through three independent measurements. For the depth-dose results, the maximum deviation of each profile to the average profile was 2.0-2.5% and 3% for the lateral dose results.

For the investigation by Raaijmakers *et al.*, a phasespace file of $50.6 \cdot 10^6$ particles was generated. The same phasespace file was imported in `GPUMCD` for the current study. The phantoms are represented as voxelised volumes of water (ρ=1.00 g/cm^3) and air (ρ=0.001205 g/cm^3). In both cases, the magnet was not part of the simulation geometry and thus only the water and air volumes of figure 5.1 were represented in the simulations.

For the depth-dose configuration, the phantom consists of 80x180x44 voxels of 1 mm^3 each.

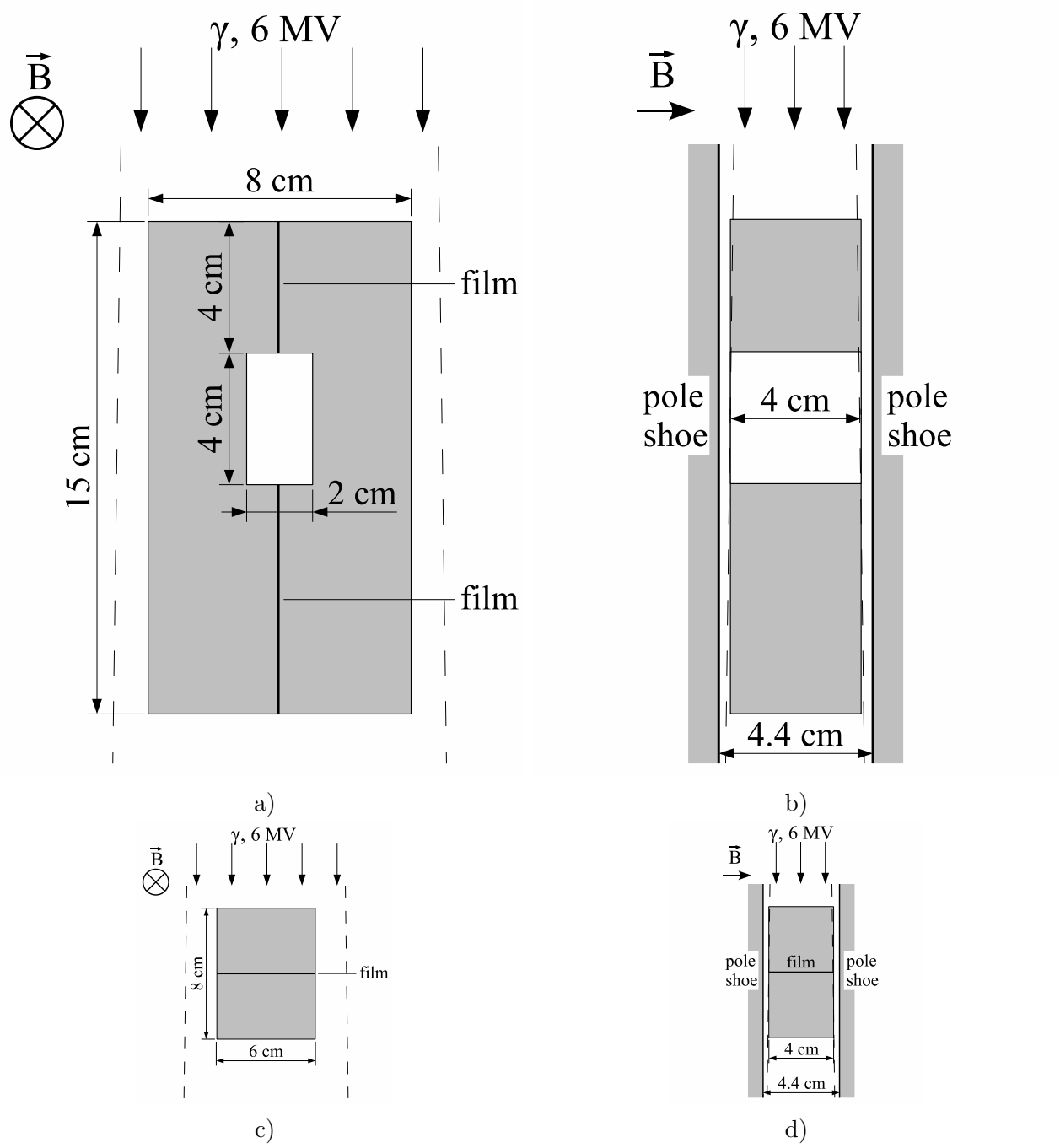

Figure 5.1 Measurement setup. Irradiation field indicated in dashed lines, PMMA phantom parts in grey. GafChromic films were positioned as indicated. (a) Depth-dose curve measurement front view. (b) Depth-dose curve measurement cross-section. (c) Lateral-dose measurement front view. (d) Lateraldose measurement crosssection.

In addition to the geometry presented in figure 5.1 a) and figure 5.1 b), 3 cm of air was added at the distal end of the water volume to allow the simulation of the electron return effect (ERE) (Raaijmakers et al., 2005).

For the lateral-dose configuration, the phantom consists of 100x90x44 voxels of 1 mm^3 each. The geometry of figure 5.1 c) and figure 5.1 d) has been left unchanged.

5.2.3 Timing evaluation

`GPUMCD` has already been extensively timed and compared to general purpose and fast Monte Carlo softwares (Hissoiny et al., 2011a). It is possible to relate to these results by evaluating the additional time required to handle magnetic fields. To do so, the execution times for the test cases described in section 5.2.2 with $B = 0$ T, $B = 0.6$ T and $B = 1.3$ T are evaluated.

Also of interest is the time required to complete one typical dose computation in the presence of a magnetic field. To complete this task, a voxelized CT-acquired prostate patient phantom is used. The phantom consists of 186x126x129 voxels of dimension 0.2x0.2x0.2 cm^3 filled with air, tissue and bone material with densities as high as 1.87 g/cm^3. Two centimeters of air have been added around the phantom to permit the tissue-air boundary ERE. A clinical beam spectrum with the correct divergence is used.

The goal of the investigation with this phantom is two-folds : to evaluate the time required for a 1-beam dose calculation in a more clinic-like situation with typical material composition and density; and to hint at the time required to calculate a 7-beams IMRT (Intensity Modulated RadioTherapy) optimization in the presence of a magnetic field. The beam size is 10x10 cm in all cases.

The multiple beams are all simulated at once with every particle's origin and direction vector sampled at the time of its generation from the user-supplied list of beam angles. This approach is definitely faster than simulating the beams individually as every beam will contribute to lower the central statistical uncertainty, therefore requiring less than 7

times the number of particles used for the 1-beam configuration to reach the same statistical uncertainty.

5.2.4 Performance evaluation

For all simulations in this investigation, the photon cutoff value was set to 10 keV and the electron cutoff value to 189 keV. The photon cross section data comes from the NIST XCOM Database (Berger et al., 1998) and the electron cross section data is generated by PEGS4. The maximum electron step, e_{s-max} in (Hissoiny et al., 2011a), was set to 0.25 mm.

All simulations of section 5.2.2 were run until the statistical uncertainty reached 1% locally at the $0.2 \cdot D_{max}$ level (See Sec. II.D of (Hissoiny et al., 2011a) for more details). For *e.g.* the depth-dose computation at 1.3 T, approximately $2.5 \cdot 10^9$ histories were simulated. The simulations of section 5.2.3 were run until the statistical uncertainty reached 1% or 2%, depending on the test, at $0.5 \cdot D_{max}$ level. This is motivated by the fact that a study by (Keall et al., 2000) shows that a statistical accuracy of less than 2% at D_{max} does not significantly impact isodoses or DVHs (Dose Value Histogram). The simulations of section 5.2.3 are therefore more precise than necessary since the average weighted error of $0.5 \cdot D_{max} < D < D_{max}$ is kept below 2%.

The dosimetric agreement is evaluated through a gamma analysis as described by Low *et al.* (Low et al., 1998). The gamma criterion is set to 2%-2mm for all comparisons.

For the comparison to experimental data in the lateral-dose case, the results were normalized so that both curves, from the experiment and from the simulation, had the average of their constant in-water dose region equal to 1.0. In the depth-dose case, the results were normalized for the area under the curve then scaled by a normalization point on the experimental curve which was not necessarily D_{max}.

All `GPUMCD` simulations were run on a PC with an Intel Core i7 980X CPU, two NVIDIA GTX480 and 6 GB of system memory. All `GPUMCD` computation times include every GPU-

CPU memory transactions required to do the simulation, including the transfer of the particles from the phase space file, which cannot fit entirely in GPU memory and is left in the system memory.

5.3 Results and Discussion

5.3.1 Experimental validation

The experimental validation results are presented in figure 5.2 for the depth-dose phantom and in figure 5.3 for the lateral dose test.

In the depth-dose test, good gamma agreement is found throughout the range for all cases except for a small region of the B=0.6 T case where the gamma test fails with gamma values of up to 1.4. In all buildup regions, at the beginning of the phantom and after the air cavity, and for all magnetic field strengths, there is a DTA (distance to agreement) of 1 voxel (1 mm). The impact of the magnetic field at B≠0 T is clearly visible in the simulation data where the buildup region is affected by the fact that electrons deposit their energy over a shorter distance. Also visible around airgaps are electrons that cannot go through the air cavity because of the ERE, resulting in a higher dose before the cavity and a second buildup after the cavity. In the first region of ERE, before the air cavity, very good agreement is found. The ERE starts at the same depth and the peaks reach the same height. Bigger differences are found in the second ERE region, at the distal end of the phantom, where the ERE starts at the same depth but the effect is not as strong in the simulation data. A DTA of 1 mm is found in that region. More air was added at the end of the phantom to verify that there is enough to permit the ERE, which did not change the results.

In the lateral-dose test, the gamma criteria is met for all points of all magnetic field strengths, except one point. The simulation volume is surrounded by air, permitting the ERE to occur at both ends of the profile. On the right side, higher dose is observed for higher values of magnetic field strengths. The opposite is found on the other side of the profile.

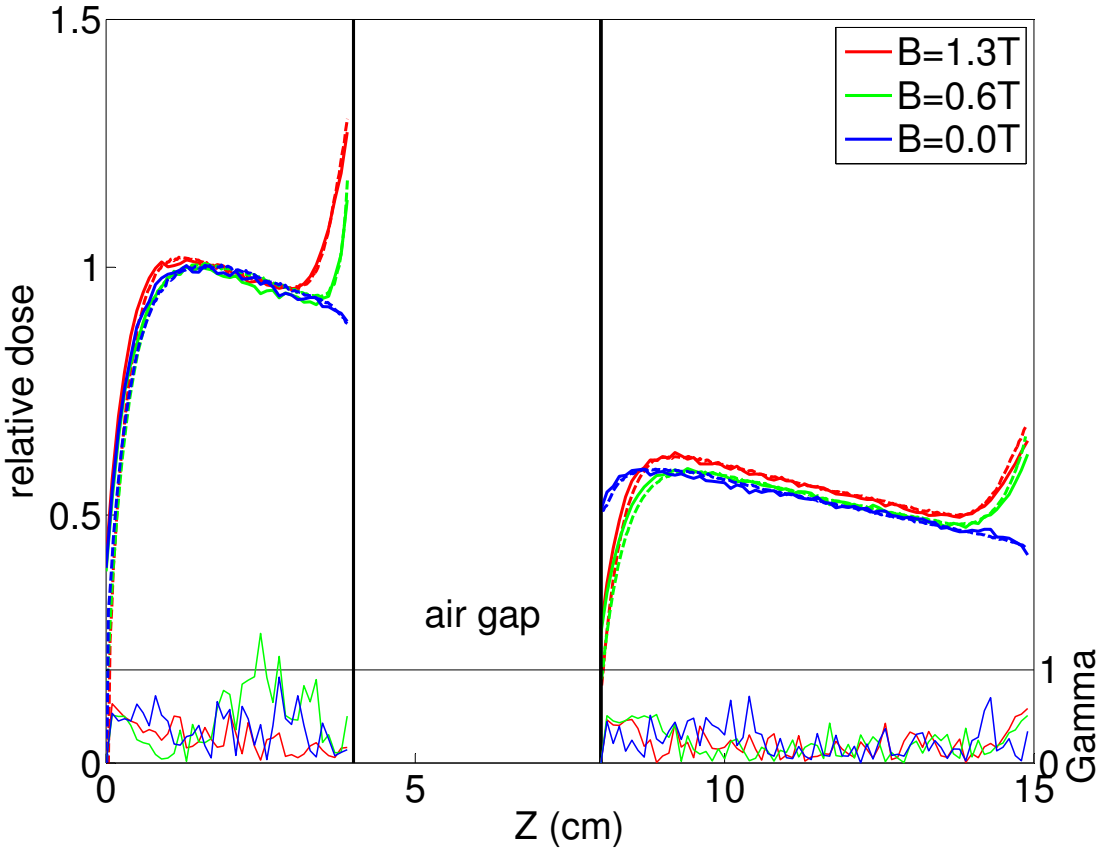

Figure 5.2 Results for the depth-dose experimental validation. The dotted lines are the experimental results and the full lines the GPUMCD simulation results. Dosimetric results as well as gamma index results with 2%-2mm criteria are presented. The results between different magnetic field strength values are not normalized equally and should therefore not be compared to each others.

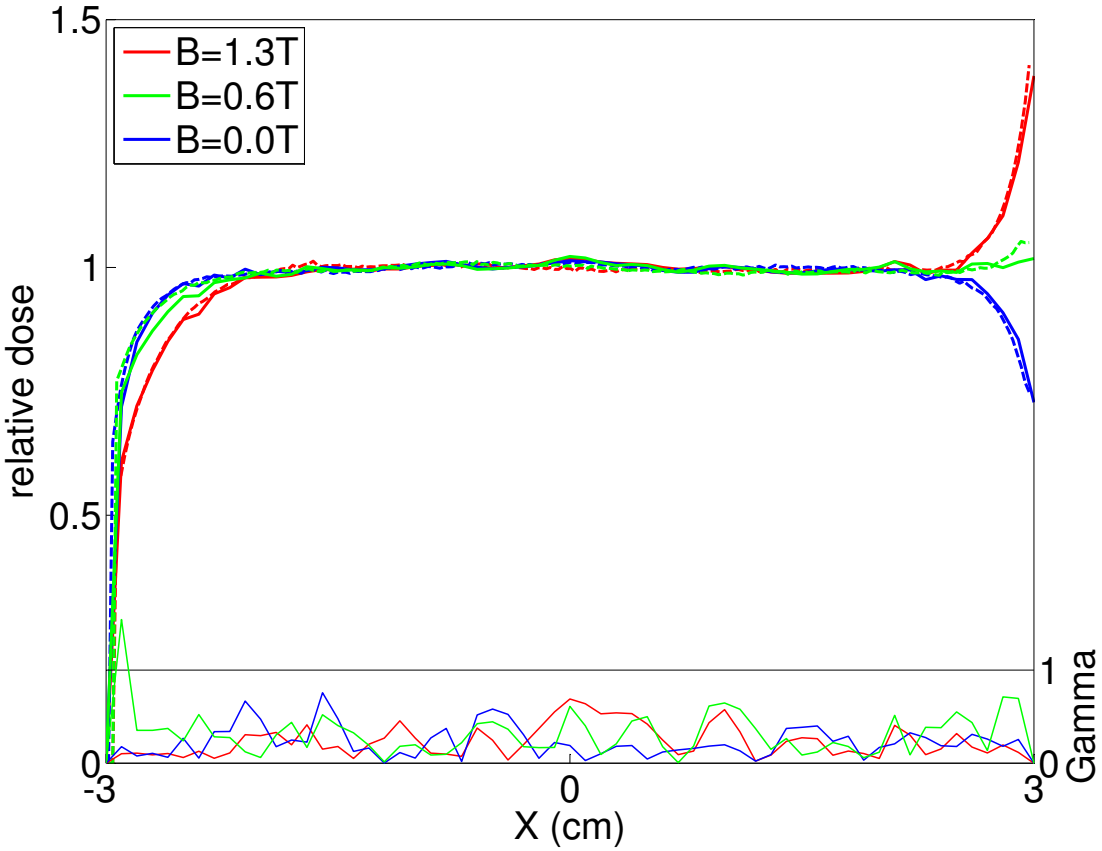

Figure 5.3 Results for the lateral-dose experimental validation. The dotted lines are the experimental results and the full lines the GPUMCD simulation results. Dosimetric results as well as gamma index results with a 2%-2mm criteria are presented. The results between different magnetic field strength values are not normalized equally and should therefore not be compared to each others.

The point where the gamma criteria fail is in the penumbra region of the 0.6 T magnetic field strength (figure 5.3). The simulation result is in between the 0 T and the 1.3 T while the experimental data puts the 0.6 T result more or less equivalent to the 0 T. As reported by Raaijmakers *et al.*, the maximum deviation when comparing each measurement to the average of all measurements was 3 %. For these two reasons, it is not impossible that the simulation result is better than the gamma index suggests.

These experimental results also represent the first experimental validation of GPUMCD.

A notable fact for both phantoms is that the gamma agreement is not only influenced by the magnetic field field strength and varies between regions with large and small magnetic field induced dose effects, even when considering B=0 T. This observation leads to the conclusion that the specific changes made to `GPUMCD` to support magnetic field fields are not the only factor in the overall agreement between the experiment and the simulation. If this were not the case, the $B = 0$ T results would present better agreement than the $B \neq 0$ T cases.

5.3.2 Timing evaluation

The time required to simulate 1 M histories for both the lateral and depth phantoms for different magnetic field strengths are reported in table 5.1.

Table 5.1 Time required for the simulation of 1 M histories in the lateral and depth phantoms for different magnetic field strengths.

	Lateral (ms)	Depth (ms)
$B = 0$ T	21.8	38.3
$B = 0.6$ T	29.5	44.1
$B = 1.3$ T	38.5	48.5

The results from table 5.1 suggest that there was a penalty for simulation in magnetic fields but that it is not directly caused by the added evaluation of equation 5.7 in the electron algorithm. If that were the case, the timing results for $B = 0.6$ T and $B = 1.3$ T would be similar. Instead, the penalty most likely comes from the added electron steps required to complete the electron path as it rotates in the ERE regions. To verify this hypothesis, a counter was placed at the calling point of the electron step algorithm. An increase of 20% in the number of calls was found when comparing $B = 0$ T and $B = 1.3$ T in the depth case.

The time required to compute the dose to 1% and 2% uncertainty (1σ) for the prostate case with 1 and 7 beams is reported in table 5.2.

The results from table 5.2 show that it is possible to compute the dose in an MRI-Linac

Table 5.2 Time required for the dose calculation of the prostate case with 1 and 7 beams to 1% and 2% uncertainty (1σ) using two GTX480. To reach 2% uncertainty in the 1-beam simulations, 100 M histories are required and 224 M histories are required in the 7-beams simulations.

	1 beam		7 beams	
	1%	2%	1%	2%
	(s)		(s)	
$B = 0$ T	31.22	7.6	41.50	10.67
$B = 0.6$ T	44.87	10.75	59.31	15.42
$B = 1.5$ T	56.77	14.99	76.90	19.88

environment using Monte Carlo techniques in a time compatible with online replanning. Times of less than 15 second are required to recompute the dose in a one beam configuration for 2% uncertainty. An increase in the computation time as the strength of the magnetic field increases is again observed here due to the fact that more electron steps are required as the electron rotates in the air region.

The 7-beams results of table 5.2 (also shown in figure 5.4) show that it is more efficient to simulate all beams in one simulation since the added execution time to simulate 7 times more beams is only approximately 1.4 times more. The reason is that what is of interest is the statistical accuracy in the voxels with $0.5 \cdot D_{max} < D < D_{max}$ which will mostly be grouped at the intersection of the 7 beams. Therefore, every beam participates to lower the statistical uncertainty. The result is of interest, if one only wishes to know the dose at the target and not at the organs at risk, as it shows that it will be more efficient to simulate 7 segments at once (one per beam) than every segment individually during a final dose calculation verification in an IMRT computation. If a low statistical uncertainty at the organs at risk is required then this observation is unusable.

5.4 Conclusion and future work

In this study, a fast and accurate dose engine for calculations in magnetic fields was presented. The validation of Monte Carlo simulations in the presence of a magnetic field using `GPUMCD`

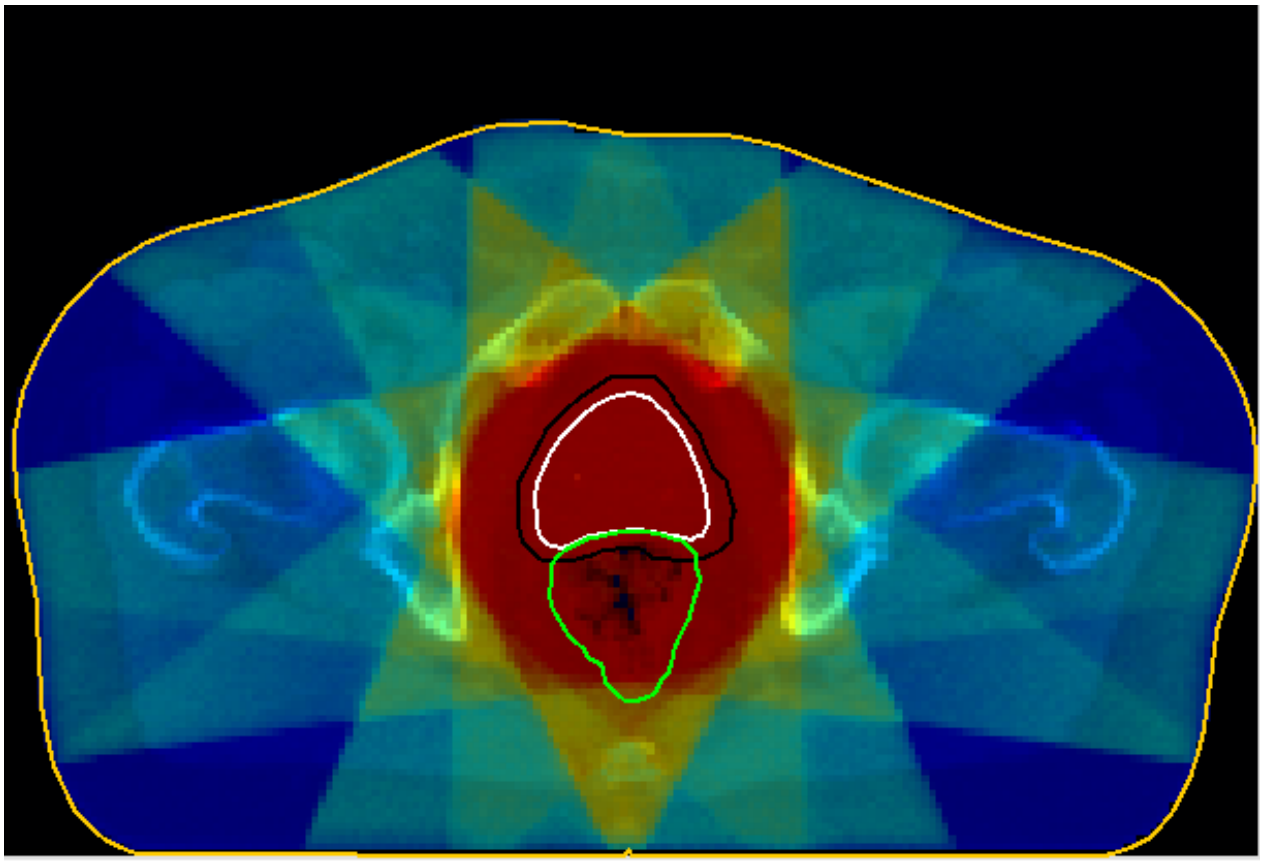

Figure 5.4 Dose results of the 7-beams simulation

has been performed in order to evaluate its accuracy to model the magnetic field effect on the computed dose as well as the time required to compute a typical dose situation.

Comparisons to experimental data showed that GPUMCD is capable of computing accurate results in strong magnetic fields, including at the boundary of air-water interfaces where the magnetic field effect is largest. Lateral-dose and depth-dose phantoms were used during the validation as well as field strengths of $B = 0$ T, $B = 0.6$ T and $B = 1.3$ T. The dosimetric results passed a gamma analysis with criteria of 2%-2 mm against the experimental data.

This work was focused on the experimental validation of GPUMCD in the presence of magnetic field. However, a brief timing study has also been performed, demonstrating that GPUMCD is able to compute the dose in a prostate patient in less than 56 second for a statistical uncertainty of 1% for one beam using two GPUs. By simulating all beams of a 7-beams plan in one simulation, less than 76 seconds are required to reach a 1% statistical uncertainty. For a 2% uncertainty, the time is reduced to 20 seconds. Four times as many GPUs could also be fitted in one workstation, bringing the time required for the computation, on a single workstation, to 5 seconds at the 2% level.

Future steps include the experimental validation of a simulation including an MLC and the magnet of the MRI. Because of the geometry of the problem, the MLC has to be placed outside the MRI and thus the use of a precomputed phase space after the magnet is not possible. The larger volume covered by the simulation as well as the presence of several high density and high-Z regions could affect the accuracy and will affect the execution time of the simulation.

Also, the time required to perform a full IMRT optimization, including the transport through the MLC and the magnet will be investigated. The 1-beams experiment results of table 5.2 can serve as a basis to evaluate the time required for inverse treatment planning in the presence of a magnetic field. IMRT optimizations require as an input a set of beamlets, discretized elements of the full field of one beam. The time required to compute the discretized beamlets will be similar to the time required for the full field. However, to generate accurate beamlets, the transport through the MLC and the magnet will have to be taken into account.

This study focuses on the time required to generate the input data and not the actual optimization. Men *et al.* have already explored GPU IMRT optimization and show that it can be accomplished in a negligible time compared to the input data generation (Men et al., 2009). A more rigourous timing study will be performed, to investigate if full online-IMRT replanning is possible with the hardware currently available.

The results from this study show that `GPUMCD` is a good candidate for online replanning dose computations for an hybrid MRI-Linac modality, where the dose received by the patient has to be computed by taking into account the magnetic field present in the MRI.

Acknowledgments

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and by Technology Foundation STW, applied science division of NWO.

CHAPITRE 6

ASPECTS MÉTHODOLOGIQUES

Plusieurs aspects informatiques ont été évacués des publications pour en alléger le contenu. Ce chapitre fait donc état d'un ensemble de considérations informatiques qui n'ont pas trouvé place dans une des publications formant cette thèse.

6.1 Architecture logicielle et options d'exécution

La simulation de Monte Carlo pour le transport de particules est un processus essentiellement séquentiel : des particules sont crées, elles sont transportées à travers la géométrie jusqu'à ce qu'elles quittent la géométrie ou tombent sous un seuil énergétique, puis des quantités d'intérêt sont sauvegardées.

Cette description amène à une architecture logicielle essentiellement procédurale. Lors de l'exécution du programme, aucune interaction n'est possible avec l'utilisateur. Seul le fichier de réglages de simulation permet d'interagir avec l'application. Le schéma global du flot d'exécution est présenté à la Fig. 6.1.

La boîte « Init » se divise en un ensemble d'opérations nécessaires pour fins d'initialisation de données utilisées par la simulation :

1. initialisation du générateur de nombres pseudo-aléatoires ;
2. lecture du fichier de réglages de la simulation ;
3. lecture des données de sections efficaces pour les matériaux présents dans la simulation et mise en texture de ces données ;
4. création de la source de particules ;
5. lecture de la géométrie dans laquelle les particules évolueront.

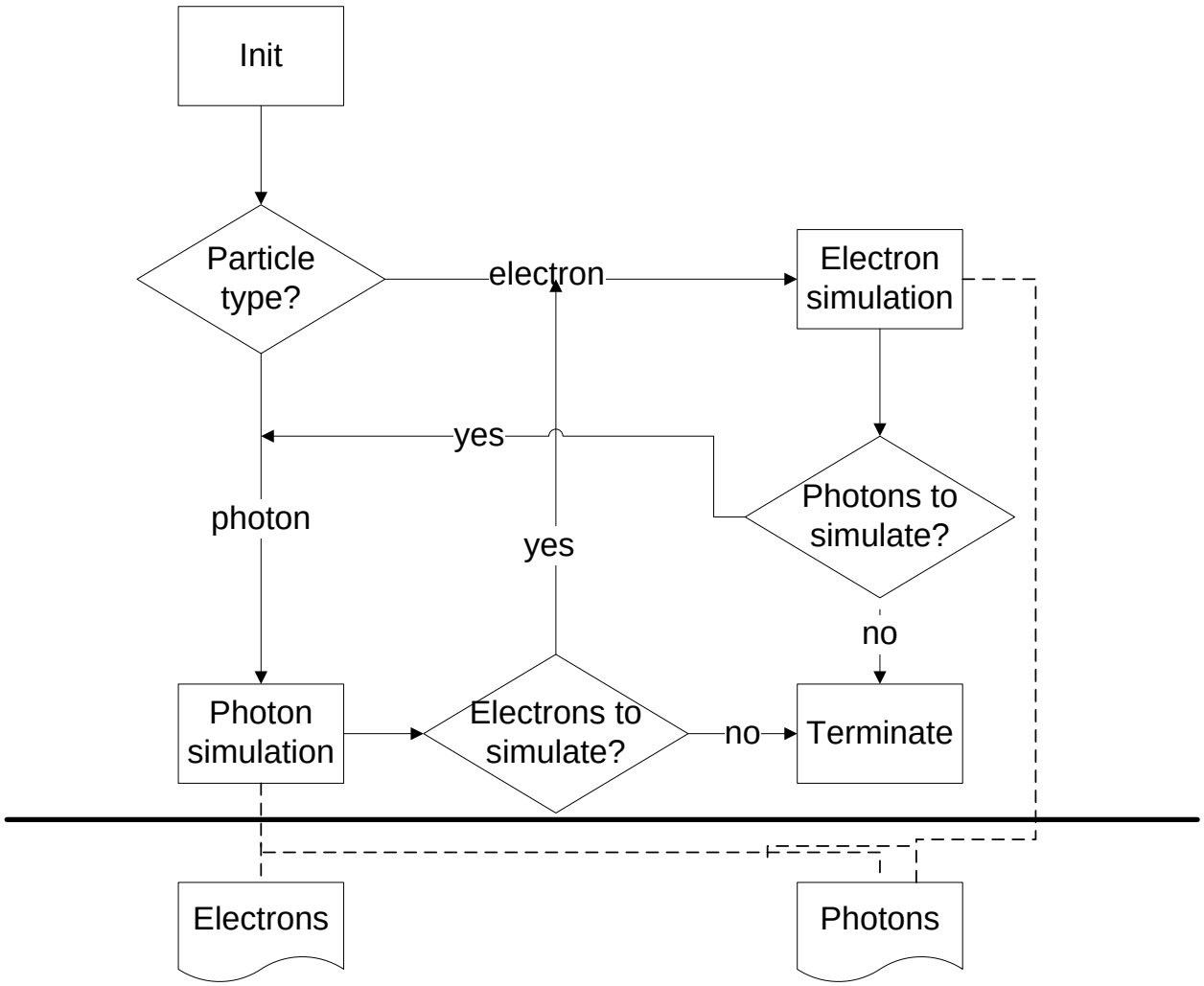

FIGURE 6.1 Schéma global du flot d'exécution dans GPUMCD.

L'annexe III (manuscrit non publié) décrit plus en détail le générateur de nombre pseudo-aléatoires utilisé.

6.1.1 Options d'exécution

La lecture du fichier de réglages donne accès aux divers types de simulations présents dans GPUMCD. Ceux-ci sont représentés à la figure 6.2.

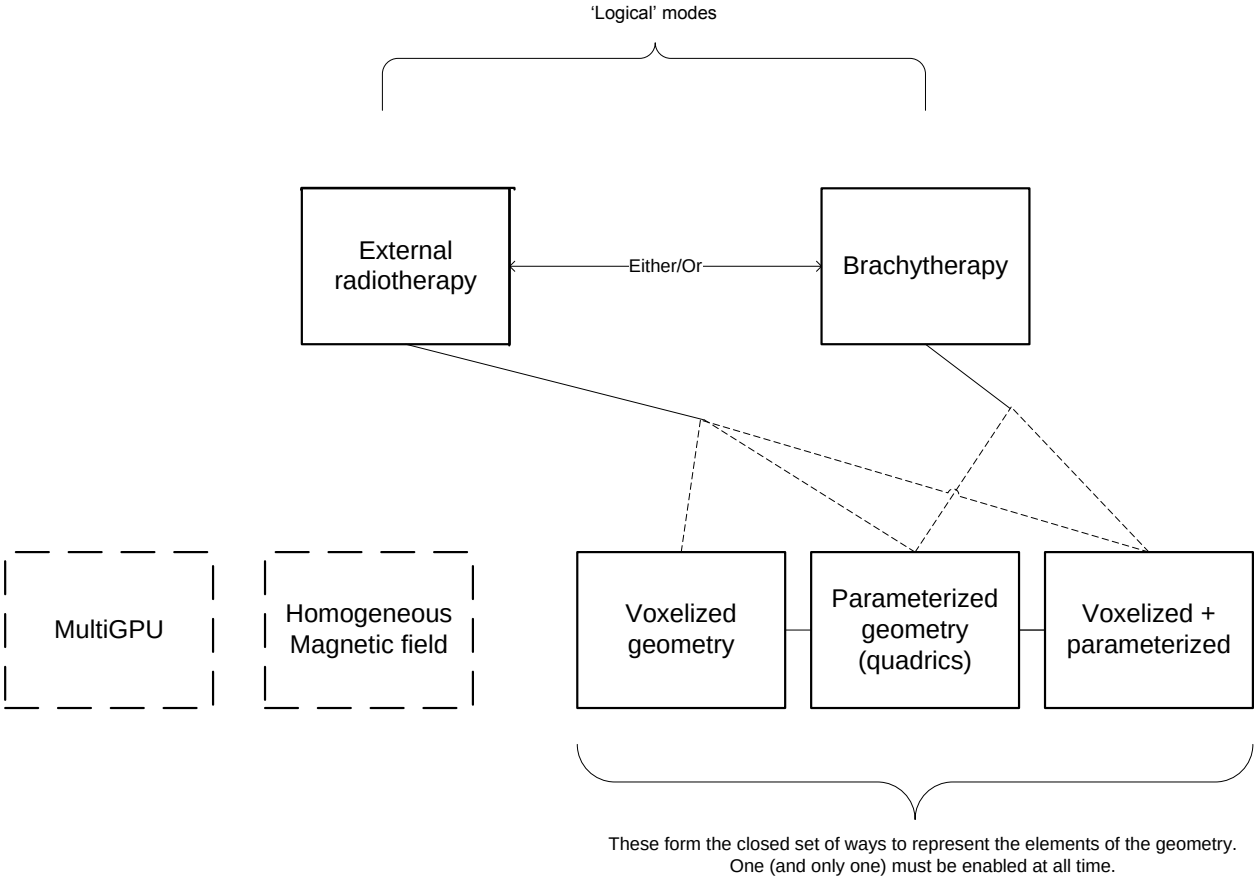

FIGURE 6.2 Options de simulation possibles dans GPUMCD.

Les traits pleins représentent une option nécessaire (parfois une parmi plusieurs) pour l'exécution d'une simulation alors que les traits pointillés représentent des options qui peuvent être utilisées ou non.

La différence entre « External radiotherapy » et « Brachytherapy » se situe surtout au niveau

des routines de comptages de valeurs d'intérêt et n'introduit pas de nouvelle physique ou de nouvelles fonctionnalités au point de vue de la simulation. Ce choix influence par contre sur les sources de particules utilisables lors de la simulation. Les sources en curiethérapie sont des entités avec une géométrie et une distribution des sites de génération de particules complexes qui n'ont pas leur équivalent en radiothérapie externe.

Le mode « MultiGPU » permet d'utiliser plus d'une carte graphique à la fois sur un même ordinateur. La division du travail se fait de façon triviale en simulant N/g particules sur chaque GPU, où N est le nombre de particules choisi par l'utilisateur et g le nombre de GPU présent. Toutes les données d'entrée sont dupliquées puisque celles-ci doivent être présentes dans la mémoire de chacune des cartes graphiques.

Le mode « Homogeneous Magnetic Field » permet à l'utilisateur de définir un champ magnétique homogène à toute la simulation, sous la forme d'un vecteur à trois dimensions. Cette option changera la routine de transport d'électrons pour incorporer l'effet des forces de Lorentz sur la trajectoire des particules chargées.

Le bloc de choix de géométrie établit comment l'univers de simulation est défini. L'option « voxelized » définit l'univers à l'aide d'un volume de voxels, chacun ayant le même volume. Le mode « parameterized » permet de définir l'univers à l'aide de surfaces géométriques du deuxième degré ainsi que par des boîtes. Finalement, le mode « Voxelized+parameterized » permet l'utilisation des deux autres options au même moment.

6.1.2 Architecture logicielle

Comme mentionné précédemment, la nature du problème laisse place à une architecture de type procédurale. Le langage choisi est le C/C++. La division des blocs logiques se fait sous la base de fichiers et de répertoires. Les grandes divisions sont :

1. Entrées / sorties
2. Initialisation de simulation

3. Sources de particules
4. Génération de nombres pseudo-aléatoires
5. Géométrie
6. Simulation de photons
7. Simulations d'électrons
8. Comptage et statistiques

La seule division énoncée ci-haut utilisant une approche orientée objet est la représentation des sources de particules. Celles-ci héritent toutes d'une classe `particleSource` virtuelle pure décrivant un prototype de source ainsi que la fonction `generateParticles` nécessaire à chaque type de source. La hiérarchie se spécialise en sources internes ou externes, à base de fichiers d'espace de phase ou de modèle analytique, à un ou plusieurs faisceaux, etc.

6.2 Accélération du transport de particules

6.2.1 Algorithme de Woodcock modifié

L'algorithme de Woodcock est une technique d'échantillonnage par rejet qui traite tout le volume modélisé comme étant constitué du matériau le plus atténuant présent dans le volume, ce qui a pour effet de réduire la distance parcourue à chaque pas et donc d'augmenter le nombre de pas nécessaires pour chaque particule. L'efficacité d'un échantillonnage par rejet se quantifie par la relation suivante :

$$e = \frac{f(X)}{c \cdot h(X)} \tag{6.1}$$

où $f(X)$ est la distribution difficilement échantillonnable (dans ce cas-ci, la distance à la prochaine interaction dans une géométrie hétérogène), $h(X)$ une distribution facilement échantillonnable (dans ce cas-ci, la distance à la prochaine interaction dans une géométrie homogène) et c une constante tel que pour tout X nous avons $f(X) < c \cdot h(X)$. Nous avons donc que plus $h(X)$ est loin de $f(X)$, plus le matériau choisi comme homogène est loin des

matériaux réels en chaque point et plus l'efficacité de l'algorithme de Woodcock est faible.

Pour une géométrie hétérogène mais connue à l'avance, GPUMCD permet de définir plusieurs régions Woodcock pour améliorer l'efficacité de l'algorithme. Ce concept est illustré à la Fig. 6.3.

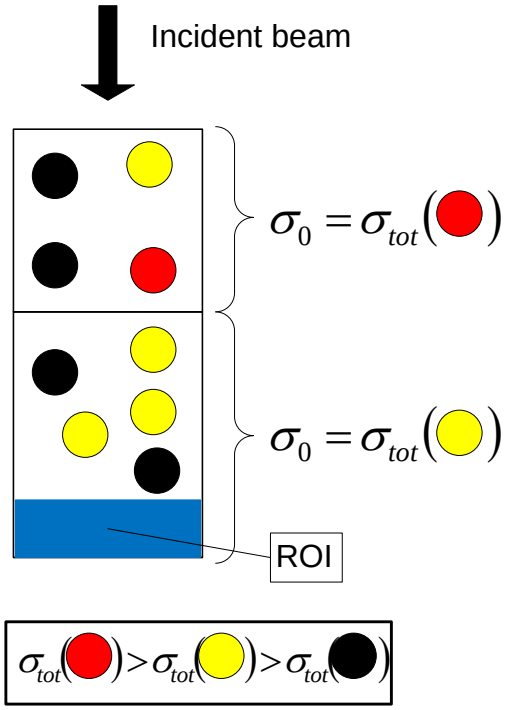

FIGURE 6.3 Algorithme de Woodcock modifié.

On y trouve deux régions de Woodcock, chacune avec sa propre section efficace maximale (la fonction $c \cdot h(X)$). Cette technique augmente l'efficacité si un sous-ensemble restreint et localisé du domaine à modéliser est très atténuant. Ce sous-ensemble, avec la technique de base, diminue l'efficacité de toute la simulation. En divisant le domaine en deux régions, le sous-domaine de faible efficacité est limité.

Cette technique ne s'applique que si la géométrie est connue d'avance et qu'elle va de *plus atténuante* à *moins atténuante* par rapport à la direction du faisceau, comme c'est le cas dans la Fig. 6.3.

Même avec cette restriction il n'est pas garanti que le transport soit exact. Une particule, arrivant de la source au haut de la figure, traversant jusqu'à la deuxième région, subissant une

forte rotation (>90 degrés) pour se trouver en direction de la première région plus atténuante n'échantillonnerait pas la longueur de son pas à partir de la bonne distribution. Pour que cette situation ait un impact, cette même particule devrait alors revenir vers la région d'intérêt (en bleu dans la Fig. 6.3). Cette technique est intéressante pour, par-exemple, le calcul d'une source de curiethérapie unique où les particules sortent du milieu dense (la source) pour aller vers le milieu moins dense (le patient) ; ou dans le cas du MRI-Linac où le faisceau doit d'abord traverser une région dense (l'imageur magnétique) avant de rejoindre une région moins dense (le patient).

6.2.2 Optimisations spécifiques à l'architecture FERMI

L'architecture FERMI de NVIDIA a amené un ensemble d'amélioration au matériel disponible. Deux d'entre elles ont eu un impact direct sur GPUMCD.

La première est la présence d'opérations atomiques pour les variables à virgule flottante. Comme les quantités d'intérêt sont sommées dans un tableau global, les opérations doivent se faire de façon atomique. Avant l'arrivée de l'architecture FERMI, ces opérations devaient faire appel à un algorithme logiciel et aux opérations d'échange atomique (`AtomicExch`) pour réaliser ce que `AtomicAdd` peut maintenant réaliser en une instruction matérielle.

La deuxième est l'ajout d'un niveau de mémoire L2 de 768 kB partagée par tous les multiprocesseurs ainsi qu'un ajout à la cache de niveau de L1, passant de 16 kB à 64 kB, avec 16/48 kB de mémoire partagée gérée par le programmeur et 48/16 kB de cache L1 pour tamponner les accès à la mémoire globale. Comme GPUMCD utilise peu de mémoire partagée, la librairie est configurée pour utiliser 16 kB de mémoire partagée et 48 kB de mémoire cache. Ces deux ressources de mémoire cache supplémentaire sont un atout important pour une simulation stochastique qui accède de façon aléatoire à la mémoire globale.

CHAPITRE 7

DISCUSSION ET OUVERTURES

Le but de ce projet était « l'étude de l'utilisation du processeur graphique (GPU) pour l'implémentation d'une plateforme de calcul de dose par simulation de Monte Carlo dans un contexte de calculs en radiothérapie » (Ref. chapitre 1). Pour ce faire, une plateforme de simulation en transport de particules par la méthode de Monte Carlo a été implémentée sur le processeur graphique.

Le premier objectif de cette thèse était « l'étude de la plateforme Monte Carlo pour des cas de radiothérapies externes où des énergies particules d'énergie de 10 keV-20MeV ». Pour atteindre cet objectif, la plateforme `GPUMCD` a été développée et présentée à la communauté par un premier article (Hissoiny et al., 2011a). Cette étape a démontré des gains impressionnants par rapport à des plateformes Monte Carlo générales et rapides tout en montrant que l'exactitude restait adéquate.

Le deuxième objectif consistait en « l'étude du comportement de [`GPUMCD`] pour des situations de curiethérapie à bas débit où plusieurs sources de faible énergie (20-35 keV) sont présentes à l'interne ». Cet objectif a conduit à un module de géométrie paramétrée à l'intérieur de la plateforme qui était jusqu'alors seulement capable de fonctionner avec une géométrie voxélisée. Ce module géométrique a aussi été utilisé lors du troisième objectif et est une nécessité pour toute simulation de géométries complexes. Ce module rend la plateforme utilisable, du moins du point de vue de la représentation du domaine, à la simulation de têtes d'accélérateur, d'appareils de mesure, etc.. L'étude en tant que tel a permis de démontrer avec quelle précision `GPUMCD` est capable de reproduire des données dosimétriques de source de curiethérapie.

Une deuxième phase à cette étude, consistera en l'évaluation de la capacité de `GPUMCD` à calculer un plan de traitement réel, dans une anatomie de patient, avec un nombre et un

positionnement de sources de calibre clinique. Cette deuxième étape permettra d'établir si GPUMCD est capable de produire des résultats assez rapidement pour rendre standard le calcul Monte Carlo en curiethérapie.

Le troisième objectif visait « la sélection de la meilleure méthode pour effectuer du calcul de transport d'énergie en présence de champs magnétiques externes ainsi que la validation du calcul avec des mesures expérimentales et une plateforme de calcul générale et établie ». La plateforme établie en question ici est GEANT4 (Agostinelli et al., 2008). Cet objectif a été atteint en incorporant l'effet de la force de Lorentz sur les particules chargées lors du transport de particules. Cette étude a été validée expérimentalement. L'atteinte de cet objectif rend possible l'utilisation de GPUMCD pour fins de calcul de dose avec le MRI-Linac.

Dans le cadre de ce projet, d'autres éléments de l'écosystème du MRI-Linac ont été intégrés à GPUMCD.

Le premier élément est une généralisation du concept de « sources de particules » de la plateforme. Une nécessité pour le MRI-Linac étant de faire du calcul de dose complet, l'utilisation d'une source physiquement réaliste est nécessaire. Pour ce faire, la possibilité de générer des particules à partir d'un espace de phase a orientation variable a été ajouté à GPUMCD. Les particules primaires sont donc maintenant générées à 0 degré, elles subissent une rotation (de leur position et de leur direction) correspondant à l'angle du faisceau, et elles sont projetées vers l'espace de simulation.

Un deuxième élément est l'ajout d'une description complète de la géométrie de l'imageur par résonance magnétique. Ceci est nécessaire puisque le MRI-Linac place l'accélérateur linéaire à l'extérieur de l'IRM. Les particules sortant du linac doivent donc traverser l'IRM avant de se rendre au patient. Un transport complet doit donc être effectué pour établir les effets de l'IRM sur le faisceau. Ultimement, il ne serait pas idéal de toujours faire un transport complet à travers l'IRM pour chaque calcul de dose puisque celui-ci devient passablement lent dû à la grande taille du domaine de simulation (l'IRM a un rayon de 1 m) ainsi qu'au très grand nombre de particules perdues dans l'IRM (environ 75%). Cette modélisation permet par contre l'élaboration, ultérieurement, d'un modèle plus efficace de transport qui pourra

être validé par une simulation « complète » considérée comme la « vérité ».

Un troisième élément est la possibilité de générer des sous-faisceaux (*beamlets*). Ceux-ci sont une donnée d'entrée à un algorithme d'optimisation de traitement inverse (IMRT). Dans le cas du MRI-Linac, les sous-faisceaux doivent aussi subir l'influence de la force de Lorentz et il est donc nécessaire des les générer à partir d'une simulation Monte Carlo. `GPUMCD` a donc été modifié pour permettre la génération de ces sous-faisceaux en effectuant un balayage 2D du faisceau complet et en simulant, individuellement, chacun de ces sous-éléments du faisceau. Après quelques raffinements de cette méthode, un programme externe génère, par traçage de rayons, une liste de sous-faisceaux à simuler et envoie cette liste à `GPUMCD`. Un programme externe utilise alors le résultat de ces simulations, les sous-faisceaux, et procède à l'optimisation IMRT.

Ces éléments font de `GPUMCD` une plateforme relativement complète pour fins de calcul de dose pour le MRI-Linac. Des temps d'environ huit minutes sont nécessaires sur une station avec une seule carte graphique, avec une possibilité de huit cartes graphiques par station de travail, pour générer les sous-faisceaux. S'il s'agit d'un simple plan à faisceaux conformes, moins de 15 secondes sont nécessaires, par faisceau, pour faire le calcul de dose. Il est donc vraisemblablement possible, avec ces temps de calcul, de faire de l'optimisation IMRT en ligne, alors que le patient est à l'intérieur du MRI-Linac et qu'un changement à son anatomie est noté par l'IRM.

Le but de cette étude a été atteint. La plateforme graphique est un candidat intéressant pour le calcul de transport d'énergie par simulation de Monte Carlo. Nous ne pouvons conclure que par rapport à l'application que nous en avons faite dans le cadre de cette thèse : le calcul de dose rapide. Il n'est pas acquis que l'implémentation d'une plateforme plus générale et exacte amènerait le même genre de temps d'exécution. En effet, plus la plateforme est exacte, plus le traitement de chaque interaction est exigeant et plus chaque interaction est aussi exigeante en temps de calcul plus l'effet de la divergence SIMD devient important.

Les principales contributions de cette thèse sont au point de vue de l'approche à utiliser pour effectuer un calcul de dose par simulation Monte Carlo sur GPU. Le traitement différé des

particules secondaires ainsi que la séparation des simulations en tâches uniformes, photon ou électron, ont permis de mieux adapter la simulation de Monte Carlo à l'architecture particulière du processeur graphique. La mise en place d'une méthode efficace de simulation de champ magnétique a aussi donné lieu à la première plateforme étant capable de faire du calcul de dose, en temps acceptable, pour le MRI-Linac. Cette contribution répond à une problématique actuelle et pratique.

Il y a par contre encore place au développement à l'intérieur de `GPUMCD`. Nous notons ici quelques améliorations qui ajouteraient à la complétude, à l'exactitude et/ou à la vitesse de la plateforme.

Premièrement, aucune technique de réduction de variance n'est présentement utilisée. Ces techniques ont le potentiel d'accélérer passablement le calcul. Plusieurs techniques de réduction de variance existent et ont été implémentées dans les plateformes existantes de transport d'énergie. Il n'est par contre pas acquis que ces techniques seront aussi bénéfiques sur une plateforme SIMD.

Deuxièmement, pour le calcul en curiethérapie, la physique des particules de basse énergie dans `GPUMCD` est passablement rudimentaire. Il serait avantageux, au niveau de l'exactitude, d'augmenter le niveau de fidélité de la physique en ajoutant, par exemple, un algorithme de relaxation atomique complet, l'effet de l'ionisation sur les collisions inélastiques, etc..

Troisièmement, l'application de techniques de réduction de bruit a le potentiel de diminuer le nombre de particules nécessaires pour obtenir une distribution de dose avec des propriétés statistiques intéressantes. Cette diminution du nombre de particules pourrait potentiellement diminuer le temps de calcul requis, si l'étape de réduction de bruit est moins coûteuse que la réduction du nombre de particules simulées.

Finalement, pour terminer cette liste non exhaustive, des améliorations importantes au transport d'électrons pourraient être apportées, surtout concernant le comportement des électrons à l'approche des frontières de milieux et l'implémentation de la diffusion multiple (*multiple scattering*).

Il est aussi intéressant de réfléchir sur l'avenir d'une telle solution. Le calcul sur GPU est-il appelé à disparaître ? Sera-t-il remplacé par quelque chose de plus convivial ? Présentement, seules des hypothèses peuvent être émises.

Il est de l'avis de l'auteur que pour extraire le parallélisme de façon adéquate, un humain restera nécessaire pour le futur à court ou moyen terme. Il en va de même pour l'agencement des données en mémoire et les accès associés. Il semble donc peu probable que des techniques strictement de compilation viendront remplacer la programmation manuelle.

Quant à la plateforme matérielle elle-même, elle a une architecture toute indiquée pour réaliser des calculs SIMD. Il n'y a donc aucune raison évidente pour revenir au calcul sur processeur quasi-monolithique. Des avancées par les deux grandes compagnies de processeurs à usage général visent à rapprocher le GPU du CPU pour que le lien de symbiose entre les deux soit de plus en plus évident. La nouvelle famille des APU (*Accelerated Processing Units*, unité de traitement accéléré) de la compagnie AMD [1] en est un bon exemple où le GPU est sur le même circuit intégré que le CPU.

De plus en plus de langages supportent le calcul sur GPU, d'une façon ou d'une autre. PyGPU a amené la programmation GPU pour le langage Python, Cuda.Net pour la plateforme .NET, etc. De plus, des librairies visent la démocratisation de la programmation GPU sans avoir à programmer directement en CUDA. Par exemple, la librairie `thrust` (Hoberock and Bell, 2010) permet de faire des opérations vectorielles sans avoir à écrire une seule ligne de CUDA. Finalement, en juillet 2011, Microsoft a annoncé son langage C++ AMP qui permet de programmer le GPU avec un nombre d'instructions comparable à une parallélisation OpenMP.

Le calcul sur GPU semble donc là pour rester. Ce qui est en voie de changement est la manière dont on programme le GPU. À l'image du langage assembleur qui est toujours utilisé et nécessaire, le langage *e.g.* CUDA restera sans doute utile et mis à jour pour l'avenir à long terme puisque certaines tâches devront continuer à être programmées dans un langage « proche du métal ».

1. http ://sites.amd.com/us/fusion/apu/Pages/fusion.aspx

CONCLUSION

La radiothérapie est l'une des méthodes possibles de traitement du cancer. Pour éliminer les cellules cancéreuses, de la radiation ionisante est donnée au patient pour que celle-ci brise les chaînes d'ADN et stoppe la réplication des cellules cancéreuses.

Pour que le traitement soit efficace, une phase de planification de traitement doit être effectuée à l'intérieur de laquelle le traitement à être donné au patient est simulé et vérifié. Le résultat de la simulation est une distribution de dépôt de dose permettant au physicien médical de décider si la dose reçue correspondra à la prescription émise par le radio-oncologue.

La distribution de dose ne sera représentative de la réalité que si l'algorithme utilisé pour calculer le dépôt de dose est lui aussi fidèle à la réalité. Plusieurs algorithmes existent, des moins exacts aux plus exacts et aussi des plus rapides aux plus lents. La simulation de Monte Carlo se situe à l'extrême exact et lent de ce spectre. Cette méthode de calcul est restée passablement inutilisée pour le calcul de dose d'un faisceau de photons dû à son temps de calcul trop élevé pour une utilisation routinière en clinique.

La question de recherche de cette thèse était donc « *comment maximiser l'utilisation d'une plateforme GPU en vue d'améliorer la vitesse d'exécution de la simulation de Monte Carlo en transport d'énergie pour le calcul de la dose en radiothérapie ?* » et le but de la thèse était « *l'étude de l'utilisation du processeur graphique (GPU) pour l'implémentation d'une plateforme de calcul de dose par simulation de Monte Carlo dans un contexte de calculs en radiothérapie* ».

Pour répondre à la question de recherche, la plateforme GPUMCD a été créée. Cette plateforme implémente une simulation de Monte Carlo couplée électron-photon et prend en considération les interactions importantes pour la plage énergétique de la radiothérapie $E\epsilon$[10 keV,20 MeV].

Le premier objectif consistait en l'évaluation de la plateforme pour des cas de radiothérapie

externe. Une première étape de validation a donc été effectuée avec une comparaison à la plateforme EGSnrc et au programme rapide de simulation de Monte Carlo : DPM. Pour trois géométries et deux types de faisceaux (6 cas au total), GPUMCD respecte des critères gamma 2%-2mm de EGSnrc tout en étant environ 1500 fois plus rapide que EGSnrc et 300 fois plus rapide que DPM.

Le deuxième objectif visait l'étude de la précision de la plateforme pour des cas de curiethérapie interne avec sources de bas débit. L'étude s'est faite avec deux sources de curiethérapie différentes et trois paramètres du formalisme TG-43. GPUMCD s'est retrouvée à l'intérieur de 4% des résultats trouvés dans la littérature. Cette différence, qui pourrait être argumentée acceptable puisqu'il n'y a pas de critère établi d'acceptation d'un algorithme de calcul, est potentiellement due aux diverses approximations faites dans la physique des particules de basse énergie dans GPUMCD.

Le troisième objectif concernait l'applicabilité de la plateforme GPUMCD pour du calcul de dose dans le contexte de l'appareil d'imagerie-traitement MRI-Linac. L'effet du champ magnétique produit par l'IRM sur les particules a dû être ajouté à la plateforme GPUMCD. Une validation expérimentale a été faite pour valider cet ajout et GPUMCD respecte des critères gamma 2%-2mm. De plus, la vitesse d'exécution de la simulation reste acceptable avec un cas de patient réel calculable en moins de 15 secondes par faisceau.

Ainsi, cette thèse peut répondre à la question de recherche et montrer que la plateforme GPU, malgré les appréhensions initiales de la communauté, est une excellente plateforme pour le calcul de dose par simulation de Monte Carlo.

Ces résultats donnent lieu à deux constats d'importance. Premièrement, GPUMCD peut être utilisé pour faire du calcul de dose routinier vu sa grande rapidité d'exécution et son exactitude. GPUMCD est aussi actuellement la seule plateforme utilisable pour faire un calcul de dose en temps acceptable pour la modalité du MRI-Linac.

Deuxièmement, l'applicabilité de la plateforme GPU au calcul Monte Carlo indique que des plateformes plus exactes et plus matures pourraient être, du moins en parties, portées

vers une implémentation GPU. La confiance de la communauté dans la physique de ces plateformes déjà établies et l'aisance acquise avec les interfaces utilisateurs pourraient alors être conservées, menant à une plus grande adoption de la plateforme graphique comme engin de calcul.

Dans les deux cas, la possibilité de calcul de dose routinier basé sur la simulation Monte Carlo permet d'envisager la fin des algorithmes analytiques ayant à leur racine plusieurs approximations. Le calcul de dose Monte Carlo mènera à une plus grande qualité de plans de traitement, puisque ceux-ci seront plus près de ce qui sera réellement donné au patient, et donc vers une plus grande qualité de traitement. Le tout pourrait se traduire par de meilleurs résultats médicaux et moins de toxicité pour les patients.

RÉFÉRENCES

(2009). *NVIDIA CUDA Compute Unified Device Architecture Programming Guide version 2.3*. NVIDIA Corporation Inc.

Agostinelli, S., Allison, J., Amako, K., Apostolakis, J., Araujo, H., Arce, P., Asai, M., Axen, D., Banerjee, S., Barrand, G., et al. (2008). GEANT4-A simulation toolkit. *Nuclear Instruments and Methods in Physics Research.*, **506**(3), 250–303.

Ahnesjö, A., Saxner, M., and Trepp, A. (1992). A pencil beam model for photon dose calculation. *Medical physics*, **19**, 263.

Aila, T. and Laine, S. (2009). Understanding the efficiency of ray traversal on gpus. In *Proc. High-Performance Graphics 2009*.

Alerstam, E., Svensson, T., and Andersson-Engels, S. (2008). Parallel computing with graphics processing units for high-speed monte carlo simulation of photon migration. *Journal of Biomedical Optics*, **13**(6), 060504.

Anderson, J., Lorenz, C., and Travesset, A. (2008). General purpose molecular dynamics simulations fully implemented on graphics processing units. *Journal of Computational Physics*, **227**(10), 5342–5359.

Badal, A. and Badano, A. (2009). Accelerating monte carlo simulations of photon transport in a voxelized geometry using a massively parallel graphics processing unit. *Medical Physics*, **36**(11), 4878–4880.

Badal, A. and Sempau, J. (2006). A package of linux scripts for the parallelization of monte carlo simulations. *Computer Physics Communications*, **175**(6), 440 – 450. ISSN 0010-4655.

Berger, M. (1963). Monte carlo calculation of the penetration and diffusion of fast charged particles. *Methods in computational physics*, **1**, 135–215.

Berger, M., Coursey, J., Zucker, M., and Chang, J. (2000). Stopping-power and range tables for electrons, protons, and helium ions. *NIST Standard Reference Database*.

Berger, M., Coursey, J., Zucker, M., and Chang, J. (2006). ESTAR, PSTAR, and ASTAR : Computer Programs for Calculating Stopping-Power and Range Tables for Electrons, Protons, and Helium Ions (version 1.2. 3).[Online] Available : http ://physics. nist. gov/Star [2006, September 25]. National Institute of Standards and Technology, Gaithersburg, MD. *Stopping-power and range tables for electrons, protons and helium ions (physics. nist. gov/PhysRefData/Star/Text/)*.

Berger, M., Hubbell, J., Seltzer, S., Chang, J., Coursey, J., Sukumar, R., and Zucker, D. (1998). XCOM : photon cross sections database. *NIST Standard Reference Database*, **8**, 87–3597.

Bielajew, A. (2001). Fundamentals of the Monte Carlo method for neutral and charged particle transport. *University of Michigan, class notes. Available online at http ://www-personal. engin. umich. edu/˜ bielajew/MCBook/book. pdf*.

Billeter, M., Olsson, O., and Assarsson, U. (2009). Efficient stream compaction on wide SIMD many-core architectures. In *Proceedings of the 1st ACM conference on High Performance Graphics*, pages 159–166. ACM.

BOMAN, E. (2007). *Radiotherapy Forward and Inverse Problem Applying Boltzmann Transport Equation*. PhD thesis, University of Kuopio, Finlande.

Borgers, C. and Larsen, E. (1996). Asymptotic derivation of the Fermi pencil-beam approximation. *Nuclear Science and Engineering*, **123**(3).

Bouchard, H. (2010). *Étude des facteurs de perturbation de chambres d'ionisation sous conditions non standard*. PhD thesis, Université de Montréal, Canada.

Brent, R. (2004). Note on marsaglias xorshift random number generators. *Journal of Statistical Software*, **11**(5), 1–4.

Carrier, J., Beaulieu, L., Therriault-Proulx, F., and Roy, R. (2006). Impact of interseed attenuation and tissue composition for permanent prostate implants. *Medical physics*, **33**, 595.

Carter, L. L., Cashwell, E. D., and Taylor, W. M. (1972). Monte carlo sampling with continuously varying cross sections along flight paths. *Nucl. Sci. Eng.*, **48**, 403–411.

Chen, Y., Bielajew, A. F., Litzenberg, D. W., Moran, J. M., and Becchetti, F. D. (2005). Magnetic confinement of electron and photon radiotherapy dose : A monte carlo simulation with a nonuniform longitudinal magnetic field. *Medical Physics*, **32**(12), 3810–3818.

Chibani, O., Williamson, J., and Todor, D. (2005). Dosimetric effects of seed anisotropy and interseed attenuation for ^{103}Pd and ^{125}I prostate implants. *Medical physics*, **32**(8).

de Greef, M., Crezee, J., van Eijk, J. C., Pool, R., and Bel, A. (2009). Accelerated ray tracing for radiotherapy dose calculations on a gpu. *Medical Physics*, **36**(9), 4095–4102.

Després, P., Lacroix, F., and Carrier, J. (2008). Tu-ee-a4-06 : Fast drr and cbct reconstruction on gpu. volume 35, pages 2915–2915. AAPM.

Ding, G. X., Cygler, J. E., Yu, C. W., Kalach, N. I., and Daskalov, G. (2005). A comparison of electron beam dose calculation accuracy between treatment planning systems using either a pencil beam or a monte carlo algorithm. *International Journal of Radiation Oncology*Biology*Physics*, **63**(2), 622 – 633.

Dolan, J., Li, Z., and Williamson, J. F. (2006). Monte carlo and experimental dosimetry of an ^{125}I brachytherapy seed. *Medical Physics*, **33**(12), 4675–4684.

Everett, C., Cashwell, E., and Turner, G. (1971). A new method of sampling the klein-nishina probability distribution for all incident photon energies above 1 kev. Technical report, LA–4663, Los Alamos Scientific Lab., N. Mex.

Fallone, B. G., Murray, B., Rathee, S., Stanescu, T., Steciw, S., Vidakovic, S., Blosser, E., and Tymofichuk, D. (2009). First mr images obtained during megavoltage photon irradiation from a prototype integrated linac-mr system. *Medical Physics*, **36**(6), 2084–2088.

Feller, W. (1968). An introduction to probability theory and its applications. Vol. 1.

Fippel, M. (1999). Fast monte carlo dose calculation for photon beams based on the vmc electron algorithm. *Medical Physics*, **26**, 1466.

Fogal, T., Childs, H., Shankar, S., Kruger, J., Bergeron, R., and Hatcher, P. (2010). Large data visualization on distributed memory multi-gpu clusters. In *Proceedings of the Conference on High Performance Graphics*, pages 57–66. Eurographics Association.

Fortin, A. (1996). *Analyse numérique pour ingénieurs*. Presses internationales polytechnique.

Gardner, J., Siebers, J., and Kawrakow, I. (2007). Dose calculation validation of vmc++ for photon beams. *Medical Physics*, **34**(5), 1809–1818.

Garland, M., Le Grand, S., Nickolls, J., Anderson, J. A., Hardwick, J., Morton, S., Phillips, E., Zhang, Y., and Volkov, V. (2008). Parallel computing experiences with cuda. *Micro, IEEE*, **28**(4), 13–27. ISSN 0272-1732.

Gearheart, D. M., Drogin, A., Sowards, K., Meigooni, A. S., and Ibbott, G. S. (2000).

Dosimetric characteristics of a new ^{125}I brachytherapy source. *Medical Physics*, **27**(10), 2278–2285.

Hensel, H., Iza-Teran, R., and Siedow, N. (2006). Deterministic model for dose calculation in photon radiotherapy. *Physics in medicine and biology*, **51**, 675.

Hirayama, H., Namito, Y., Nelson, W., Bielajew, A., Wilderman, S., and Michigan, U. (2005). *The EGS5 code system*. United States. Dept. of Energy.

Hissoiny, S., Ozell, B., Bouchard, H., and Després, P. (2011a). Gpumcd : A new gpu-oriented monte carlo dose calculation platform. *Medical Physics*, **38**(2), 754–764.

Hissoiny, S., Ozell, B., and Després, P. (2010a). A convolution-superposition dose calculation engine for gpus. *Medical Physics*, **37**(3), 1029–1037.

Hissoiny, S., Ozell, B., Després, P., and Carrier, J.-F. (2011b). Validation of gpumcd for low-energy brachytherapy seed dosimetry. *Medical Physics*, **38**(7), 4101–4107.

Hissoiny, S., Ozell, B., and Després, P. (2009). Fast convolution-superposition dose calculation on graphics hardware. *Medical Physics*, **36**(6), 1998–2005.

Hissoiny, S., Ozell, B., and Després, P. (2010b). Gpumcd, a new gpu-oriented monte carlo dose calculation platform. In *XVIth ICCR Conference Proceedings*. Amsterdam, NL.

Hissoiny, S., Raaijmakers, A. J. E., Ozell, B., Després, P., and Raaymakers, B. W. (2011c). Fast dose calculation in magnetic fields with gpumcd. *Physics in Medicine and Biology*, **56**(16), 5119.

Hoberock, J. and Bell, N. (2010). Thrust : A parallel template library. Version 1.3.0.

Ibbott, G. S., Meigooni, A. S., and Gearheart, D. M. (2002). Monte carlo determination of dose rate constant. *Medical Physics*, **29**(7), 1637–1638.

Ibbott, G. S. and Nath, R. (2001). Dose-rate constant for imagyn ^{125}I brachytherapy source. *Medical Physics*, **28**(4), 705–705.

James, F. (1980). Monte carlo theory and practice. *Reports on Progress in Physics*, **43**(9), 1145.

Janowczyk, A., Chandran, S., and Aluru, S. (2008). Fast, processor-cardinality agnostic prng with a tracking application. In *Computer Vision, Graphics and Image Processing, 2008. ICVGIP 08. Sixth Indian Conference on*, pages 171–178. ISSN.

Jeleń, U., Sohn, M., and Alber, M. (2005). A finite size pencil beam for IMRT dose optimization. *Physics in Medicine and Biology*, **50**, 1747.

Jia, X., Gu, X., Sempau, J., Choi, D., Majumdar, A., and Jiang, S. B. (2010a). Development of a gpu based monte carlo dose calculation code for coupled electron photon transport. *Physics in Medicine and Biology*, **55**(11), 3077.

Jia, X., Lou, Y., Li, R., Song, W., and Jiang, S. (2010b). GPU-based fast cone beam CT reconstruction from undersampled and noisy projection data via total variation. *Medical physics*, **37**, 1757.

Johns, H. E. and Cunningham, J. R. (1983). *The Physics of Radiology, fourth edition*, page 181. Charles C Thomas, Springfield IL.

Kawrakow, I. and Bielajew, A. (1998). On the representation of electron multiple elastic-scattering distributions for monte carlo calculations. *Nuclear Instruments and Methods in Physics Research Section B : Beam Interactions with Materials and Atoms*, **134**(3), 325–336.

Kawrakow, I. and Fippel, M. (2000). Investigation of variance reduction techniques for monte carlo photon dose calculation using xvmc. *Physics in Medicine and Biology*, **45**(8), 2163–2184.

Kawrakow, I., Fippel, M., and Friedrich, K. (1996). 3d electron dose calculation using a voxel based monte carlo algorithm (vmc). *Medical physics*, **23**, 445.

Kawrakow, I. and Rogers, D. (2000). The EGSnrc code system : Monte Carlo simulation of electron and photon transport. *NRCC Report*.

Kawrakow, I. and Rogers, D. (2003). PIRS-701 : The EGSnrc Code System : Monte Carlo Simulation of Electron and Photon Transport. *Ionizing Radiation Standards (NRC, Ottawa, Ontario, 2003)*.

Keall, P. J., Siebers, J. V., Jeraj, R., and Mohan, R. (2000). The effect of dose calculation uncertainty on the evaluation of radiotherapy plans. *Medical Physics*, **27**(3), 478–484.

King, G., Cai, Z., Lu, Y., Wu, J., Shih, H., and Chang, C. (2010). A high-performance multi-user service system for financial analytics based on web service and gpu computation. In *Parallel and Distributed Processing with Applications (ISPA), 2010 International Symposium on*, pages 327–333. IEEE.

Kirkby, C., Stanescu, T., Rathee, S., Carlone, M., Murray, B., and Fallone, B. G. (2008). Patient dosimetry for hybrid mri-radiotherapy systems. *Medical Physics*, **35**(3), 1019–1027.

Klein, O. and Nishina, T. (1929). Uber die streuung von strahlung durch freie elektronen nach der neuen relativistischen quantendynamik von dirac. *Zeitschrift fur Physik A Hadrons and Nuclei*, **52**(11), 853–868.

Koch, H. and Motz, J. (1959). Bremsstrahlung cross-section formulas and related data.

Reviews of Modern Physics, **31**(4), 920–955.

Krieger, T. and Sauer, O. (2005). Monte Carlo-versus pencil-beam-/collapsed-cone-dose calculation in a heterogeneous multi-layer phantom. *Physics in medicine and biology*, **50**, 859.

Kutter, O., Karamalis, A., Wein, W., and Navab, N. (2009a). A gpu-based framework for simulation of medical ultrasound. volume 7261, page 726117. SPIE.

Kutter, O., Shams, R., and Navab, N. (2009b). Visualization and GPU-accelerated simulation of medical ultrasound from CT images. *Computer methods and programs in biomedicine*, **94**(3), 250–266.

Lagendijk, J., Raaymakers, B., Raaijmakers, A., Overweg, J., Brown, K., Kerkhof, E., van der Put, R., Hårdemark, B., van Vulpen, M., and van der Heide, U. (2008). MRI/linac integration. *Radiotherapy and oncology*, **86**(1), 25.

Langdon, B. (2009). A fast high quality pseudo random number generator for nvidia cuda. GECCO 2009 Workshop, Tutorial and Competition on Computational Intelligence on Consumer Games and Graphics Hardware CIGPU.

L'ecuyer, P. (1988). Efficient and portable combined random number generators. *Commun. ACM*, **31**(6), 742–751. ISSN 0001-0782.

L'ecuyer, P. (1999). Tables of linear congruential generators of different sizes and good lattice structure. *Math. Comput.*, **68**(225), 249–260. ISSN 0025-5718.

L'ecuyer, P. and Simard, R. (2007). Testu01 : A c library for empirical testing of random number generators. *ACM Trans. Math. Softw.*, **33**(4), 22.

Lee, Y. K., Bollet, M., Charles-Edwards, G., Flower, M. A., Leach, M. O., McNair, H., Moore, E., Rowbottom, C., and Webb, S. (2003). Radiotherapy treatment planning of prostate cancer using magnetic resonance imaging alone. *Radiotherapy and Oncology*, **66**(2), 203 – 216.

Leppanen, J. (2007). Development of a New Monte Carlo reactor physics code. *VTT PUBLICATIONS*, **640**.

Lindley, C. (1992). *Practical ray tracing in C*. John Wiley Sons.

Liu, H., Mackie, T., and McCullough, E. (1997). A dual source photon beam model used in convolution/superposition dose calculations for clinical megavoltage x-ray beams. *Medical Physics*, **24**, 1960.

Low, D. and Dempsey, J. (2003). Evaluation of the gamma dose distribution comparison method. *Medical Physics*, **30**, 2455.

Low, D. A., Harms, W. B., Mutic, S., and Purdy, J. A. (1998). A technique for the quantitative evaluation of dose distributions. *Medical Physics*, **25**(5), 656–661.

Ma, C.-M., Pawlicki, T., Jiang, S. B., Li, J. S., Deng, J., Mok, E., Kapur, A., Xing, L., Ma, L., and Boyer, A. L. (2000). Monte carlo verification of imrt dose distributions from a commercial treatment planning optimization system. *Physics in Medicine and Biology*, **45**(9), 2483.

Mackie, T., Scrimger, J., and Battista, J. (1985). A convolution method of calculating dose for 15-MV x rays. *Medical physics*, **12**, 188.

Marsaglia, G. (1995). Diehard, a battery of tests for random number generators.

Marsaglia, G. (1997). A random number generator for C. Posted to the `sci.math.num-analysis` news group.

Marsaglia, G. (2003a). Random number generation. In *Encyclopedia of Computer Science*, pages 1499–1503. John Wiley and Sons Ltd., Chichester, UK. ISBN 0-470-86412-5.

Marsaglia, G. (2003b). Xorshift rngs. *Journal of Statistical Software*, **8**(14).

Marsaglia, G. and Zaman, A. (1991). A new class of random number generators. *The Annals of Applied Probability*, **1**(3), 462.480.

Matsumoto, M. and Nishimura, T. (1998). Mersenne twister : a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, **8**(1), 3–30. ISSN 1049-3301.

Men, C., Gu, X., Choi, D., Majumdar, A., Zheng, Z., Mueller, K., and Jiang, S. B. (2009). Gpu-based ultrafast imrt plan optimization. *Physics in Medicine and Biology*, **54**(21), 6565.

Micikevicius, P. (2009). 3d finite difference computation on gpus using cuda. In *GPGPU-2 : Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units*, New York, NY, USA, pages 79–84. ACM.

Mobit, P. and Badragan, I. (2004). Dose perturbation effects in prostate seed implant brachytherapy with ^{125}I. *Physics in Medicine and Biology*, **49**, 3171.

Mohan, R., Chui, C., and Lidofsky, L. (1986). Differential pencil beam dose computation model for photons. *Medical Physics*, **13**(1), 64–73.

Møller, C. (1932). Zur theorie des durchgangs schneller elektronen durch materie. *Annalen der Physik*, **406**, 531–585.

Nath, R. and Yue, N. (2000). Dose distribution along the transverse axis of a new ^{125}I source for interstitial brachytherapy. *Medical Physics*, **27**(11), 2536–2540.

Nelson, W., Hirayama, H., and Rogers, D. (1985). The EGS4 code system. *SLAC-265, Stanford Linear Accelerator Center*.

Nvidia, C. (2010). Compute Unified Device Architecture Programming Guide. *NVIDIA : Santa Clara, CA*.

Nyland, L., Harris, M., and Prins, J. (2007). Fast n-body simulation with cuda. In Nguyen, H., editor, *GPU Gems 3*, chapter 31. Addison Wesley Professional.

Panneton, F. and L'Ecuyer, P. (2005). On the xorshift random number generators. *ACM Trans. Model. Comput. Simul.*, **15**(4), 346–361. ISSN 1049-3301.

Park, S. K. and Miller, K. W. (1988). Random number generators : good ones are hard to find. *Commun. ACM*, **31**(10), 1192–1201. ISSN 0001-0782.

Plechaty, E., Cullen, D., and Howerton, R. (1978). Tables and graphs of photon-interaction cross sections from 0. 1 keV to 100 MeV derived from the LLL Evaluated-Nuclear-Data Library. Technical Report Report No. UCRL-50400, Vol. 6, Rev. 2, Lawrence Livermore Laboratory, Livermore, CA.

Podlozhnyuk, V. (2007). *Parallel Mersenne Twister*. NVIDIA Corporation Inc.

Raaijmakers, A. J. E., Raaymakers, B. W., and Lagendijk, J. J. W. (2005). Integrating a mri scanner with a 6 mv radiotherapy accelerator : dose increase at tissue-air interfaces in a lateral magnetic field due to returning electrons. *Physics in Medicine and Biology*, **50**(7), 1363.

Raaijmakers, A. J. E., Raaymakers, B. W., and Lagendijk, J. J. W. (2007). Experimental verification of magnetic field dose effects for the mri-accelerator. *Physics in Medicine and Biology*, **52**(14), 4283.

Raaijmakers, A. J. E., Raaymakers, B. W., and Lagendijk, J. J. W. (2008). Magnetic-field-induced dose effects in mr-guided radiotherapy systems : dependence on the magnetic field strength. *Physics in Medicine and Biology*, **53**(4), 909.

Raaymakers, B. W., Lagendijk, J. J. W., Overweg, J., Kok, J. G. M., Raaijmakers, A. J. E., Kerkhof, E. M., van der Put, R. W., Meijsing, I., Crijns, S. P. M., Benedosso, F., van Vulpen, M., de Graaff, C. H. W., Allen, J., and Brown, K. J. (2009). Integrating a 1.5 t mri scanner with a 6 mv accelerator : proof of concept. *Physics in Medicine and Biology*, **54**(12), N229.

Rivard, M., Coursey, B., DeWerd, L., Hanson, W., Huq, M., Ibbott, G., Mitch, M., Nath, R., and Williamson, J. (2004). Update of AAPM Task Group No. 43 Report : A revised AAPM protocol for brachytherapy dose calculations. *Medical Physics*, **31**, 633.

Rivard, M. J. (2001). Monte carlo calculations of aapm task group report no. 43 dosimetry parameters for the med3631-a/m ^{125}I source. *Medical Physics*, **28**(4), 629–637.

Rogers, D. (2006). Fifty years of Monte Carlo simulations for medical physics. *Physics in medicine and biology*, **51**, R287.

Salvat, F., Fern'andez-Varea, J., and Sempau, J. (2003). Penelope a code system for monte carlo simulation of electron and photon transport. OECD.

Salvat, F., Fernández-Varea, J., and Sempau, J. (2006). PENELOPE-2006 : a code system for Monte Carlo simulation of electron and photon transport. In *Workshop Proceedings*, volume 4, page 7.

Satish, N., Harris, M., and Garland, M. (2009). Designing efficient sorting algorithms for manycore gpus. *Parallel and Distributed Processing Symposium, International*, **0**, 1–10.

Sauter, F. (1931).
"Uber den atomaren Photoeffekt in der K-Schale nach der relativistischen Wellenmechanik Diracs. *Annalen der Physik*, **403**, 454–488.

Schneider, J., Kraus, M., and Westermann, R. (2010). Gpu-based euclidean distance transforms and their application to volume rendering. *Computer Vision, Imaging and Computer Graphics. Theory and Applications*, pages 215–228.

Seltzer, S., Lamperti, P., Loevinger, R., Mitch, M., Weaver, J., and Coursey, B. (2003). New national air-kerma-strength standards for ^{125}I and ^{103}I brachytherapy seeds. *J. Res. Natl. Inst. Stand. Technol*, **108**, 337–358.

Sempau, J., Wilderman, S. J., and Bielajew, A. F. (2000). Dpm, a fast, accurate monte carlo code optimized for photon and electron radiotherapy treatment planning dose calculations. *Physics in Medicine and Biology*, **45**(8), 2263–2292.

Sriram, V. and Kearney, D. (2007). High throughput multi-port mt19937 uniform random number generator. *Parallel and Distributed Computing Applications and Technologies, International Conference on*, pages 157–158. ISBN 0-7695-3049-4.

Taylor, R. and Rogers, D. (2008). An EGSnrc Monte Carlo-calculated database of TG-43 parameters. *Medical physics*, **35**, 4228.

Taylor, R., Yegin, G., and Rogers, D. (2007). Benchmarking brachydose : Voxel based EGSnrc Monte Carlo calculations of TG-43 dosimetry parameters. *Medical physics*, **34**, 445.

Thomas, D. B., Howes, L., and Luk, W. (2009). A comparison of cpus, gpus, fpgas, and massively parallel processor arrays for random number generation. In *FPGA '09 : Proceeding of the ACM/SIGDA international symposium on Field programmable gate arrays*, New York, NY, USA, pages 63–72. ACM.

Thomson, R., Yegin, G., Taylor, R., Sutherland, J., and Rogers, D. (2010). Sci—sat am(2) : Brachy — 05 : Fast monte carlo dose calculations for brachytherapy with brachydose. volume 37, pages 3910–3911. AAPM.

Tomov, S., Dongarra, J., and Baboulin, M. (2010). Towards dense linear algebra for hybrid gpu accelerated manycore systems. *Parallel Computing*, **36**(5-6), 232–240.

Van Dyk, J. et al. (1999). *The modern technology of radiation oncology*, pages 252–253. Medical Physics Publ.

Veach, E. (1997). *Robust Monte Carlo methods for light transport simulation*. PhD thesis, Stanford University, États-Unis.

Volkov, V. and Demmel, J. W. (2008). Benchmarking gpus to tune dense linear algebra. In *SC '08 : Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, Piscataway, NJ, USA, pages 1–11. IEEE Press.

Wang, L. L. W., Jacques, S., and Zheng, L. (1995). Mcml monte carlo modeling of light transport in multi-layered tissues. *Computer methods and programs in biomedicine*, **47**(2), 131–146.

Wang, P., Abel, T., and Kaehler, R. (2010). Adaptive mesh fluid simulations on gpu. *New Astronomy*, **15**(7), 581–589.

Wang, R., Wang, R., Zhou, K., Pan, M., and Bao, H. (2009). An efficient gpu-based approach

for interactive global illumination. In *SIGGRAPH '09 : ACM SIGGRAPH 2009 papers*, New York, NY, USA, pages 1–8. ACM.

Williamson, J. and Quintero, F. (1988). Theoretical evaluation of dose distributions in water about models 6711 and 6702 ^{125}I seeds. *Medical Physics*, **15**, 891–897.

Williamson, J. F. (2002). Dosimetric characteristics of the draximage model ls-1 ^{125}I interstitial brachytherapy source design : A monte carlo investigation. *Medical Physics*, **29**(4), 509–521.

Woo, A. (1990). Fast ray-box intersection. In *Graphics Gems*, page 396. Academic Press Professional, Inc.

Woodcock, E., Murphy, T., Hemmings, P., and Longworth, S. (1965). Techniques used in the gem code for monte carlo neutronics calculations in reactors and other systems of complex geometry. In *Proceedings of the Conference on Applications of Computing Methods to Reactor Problems*, page 557.

Wulff, J., Zink, K., and Kawrakow, I. (2008). Efficiency improvements for ion chamber calculations in high energy photon beams. *Medical physics*, **35**, 1328.

ANNEXE I

DÉRIVATION DES ÉQUATIONS DE TRANSPORT DE PARTICULES EN PRÉSENCE DE CHAMP MAGNÉTIQUE HOMOGÈNE

Le point de départ de la dérivation des équations de transport de particules en présence de champ magnétique est l'équation des forces de Lorentz :

$$\vec{F} = \frac{d\vec{p}}{dt} = q(\vec{E} + \frac{\vec{v}}{c} \times \vec{B}) \tag{I.1}$$

où $\vec{E}$ représente le champ électrique et $\vec{B}$ le champ magnétique, $\vec{v}$ est la vitesse de la particule et q sa charge. Pour le cas qui nous occupe, nous nous intéressons aux électrons ($q = e$) et aux champs magnétique homogène ($\vec{E} = 0$). On trouve donc

$$\frac{md(\gamma\vec{v})}{dt} = e(\frac{\vec{v}}{c} \times \vec{B}) \tag{I.2}$$

où l'identité $\vec{p} = \gamma m\vec{v}$ a été utilisée et $\gamma = \frac{1}{\sqrt{1-\frac{v^2}{c^2}}}$. L'opérateur de gauche peut s'exprimer selon l'expression suivante

$$\frac{md\gamma\vec{v}}{dt} = \frac{\gamma^3 mv}{c^2}\frac{dv}{dt}\vec{v} + \gamma m\frac{d\vec{v}}{dt}, \tag{I.3}$$

en suivant le développement suivant (où le terme m a été ignoré pour simplifier l'écriture et où $v = |\vec{v}|$) :

$$\begin{aligned}\frac{d\gamma\vec{v}}{dt} &= \frac{d\gamma}{dt}\vec{v} + \gamma\frac{d\vec{v}}{dt} = \frac{d\left[\left(1-\frac{v^2}{c^2}\right)^{-1/2}\right]}{dt}\vec{v} + \gamma\frac{d\vec{v}}{dt} \\ &= \left(\frac{-1}{2}\right)\left(1-\frac{v^2}{c^2}\right)^{-3/2}\left(\frac{-2v}{c^2}\frac{dv}{dt}\right)\vec{v} + \gamma\frac{d\vec{v}}{dt} \\ &= (1-\beta^2)^{-3/2}\left(\frac{v}{c^2}\frac{dv}{dt}\right)\vec{v} + \gamma\frac{d\vec{v}}{dt} \\ &= \gamma^3\frac{v}{c^2}\frac{dv}{dt}\vec{v} + \gamma\frac{d\vec{v}}{dt}\end{aligned} \tag{I.4}$$

où les identités $\gamma = 1/\sqrt{1-\beta^2}$ et $\beta = v/c$ ont été utilisées. En réinjectant dans Eq. I.2, on trouve

$$\frac{\gamma^3 mv}{c^2}\frac{dv}{dt}\vec{v} + \gamma m\frac{d\vec{v}}{dt} = e(\frac{\vec{v}}{c}\times\vec{B}), \tag{I.5}$$

réécrite sous la forme :

$$\begin{aligned}\frac{e}{m}(\beta\hat{v}\times\vec{B}) &= \frac{\gamma^3 v}{c^2}\frac{dv}{dt}\vec{v} + \gamma\frac{d\vec{v}}{dt} \\ &= \frac{\gamma^3(c\beta)}{c^2}\frac{cd\beta}{dt}\beta c\hat{v} + \gamma c\beta\frac{d\hat{v}}{dt} \\ \frac{e}{mc}(\beta\hat{v}\times\vec{B}) &= \gamma^3\frac{d\beta}{dt}\hat{v} + \gamma\beta\frac{d\hat{v}}{dt}\end{aligned} \tag{I.6}$$

où $\hat{v} = \vec{v}/|\vec{v}|$.

Les vecteurs $\hat{v}$ et $\frac{d\hat{v}}{dt}$ sont orthogonaux puisque le mouvement des particules à l'intérieur d'un champ magnétique suit une trajectoire hélicoïdale. Il est donc possible de décomposer l'Eq. I.6 par projection sur ces deux vecteurs. La projection sur $\hat{v}$ donne

$$\begin{aligned}\left(\frac{e}{mc}(\beta\hat{v}\times\vec{B})\right)\hat{v} &= \left(\gamma\beta\frac{d\hat{v}}{dt} + \gamma^3\frac{d\beta}{dt}\hat{v}\right)\hat{v} \\ 0 &= \gamma^3\frac{d\beta}{dt}\hat{v}\hat{v} \\ 0 &= \frac{d\beta}{dt} = 0.\end{aligned} \tag{I.7}$$

Cette dernière expression démontre que le module de la vitesse de la particule ne change pas, ce qui est conforme avec le fait que les champs magnétiques ne fournissent pas de travail sur les particules. La projection sur $\frac{d\hat{v}}{dt}$ se trouve par addition des deux bases :

$$\begin{aligned} 0\hat{v} + \gamma\beta\frac{d\hat{v}}{dt} &= \frac{e}{mc}(\beta\hat{v} \times \vec{B}) \\ \frac{d\hat{v}}{dt} &= \frac{e}{mc\gamma}(\hat{v} \times \vec{B}) \end{aligned} \tag{I.8}$$

Finalement, l'accélération $d\vec{v}/dt$ se trouve à être

$$a = \frac{d\vec{v}}{dt} = \frac{d(c\hat{v}\beta)}{dt} = c\left(\frac{d\beta}{dt}\hat{v} + \beta\frac{d\hat{v}}{dt}\right) = \frac{e\beta}{m\gamma}(\hat{v} \times \vec{B}) \tag{I.9}$$

À partir de cette expression il est possible de définir le changement d'orientation $\Delta\hat{v}$ que subit une particule à l'intérieur d'un champ magnétique. Cette étude sera ici faite à l'aide de l'approximation que l'accélération ne varie pas rapidement et qu'elle peut être considérée constante lors d'un saut d'électron donné. À l'aide de cette approximation, il est possible de poser

$$\Delta\vec{v} = a_0 t. \tag{I.10}$$

Puisque la simulation de Monte Carlo opère avec des sauts en distance s et non en temps t, l'expression est modifiée en utilisant encore une fois l'approximation que la vitesse est constante durant le saut

$$\Delta\vec{v} = \frac{s}{v_0}a_0 = s\frac{e(\hat{v_0} \times \vec{B})}{m\gamma_0 c}, \tag{I.11}$$

où tous les indices 0 représentent les valeurs au début du saut d'électron.

ANNEXE II

BIAIS D'ESTIMATEURS

Estimateur de moyenne L'estimateur de moyenne $\overline{X}$ est non biaisé puisque :

$$\begin{aligned} E(\overline{X}) &= E\left[\frac{1}{N}\sum_{i=1}^{N} X_i\right] \\ &= E\left[\frac{1}{N}\sum_{i=1}^{N}(X_i-\mu)+\mu\right] \\ &= \frac{1}{N}\sum_{i=1}^{N}\left[E(X_i-\mu)+E(\mu)\right] \\ &= \frac{1}{N}\left[NE(\mu)\right] \\ &= \mu \end{aligned} \tag{II.1}$$

Estimateur de variance L'estimateur de variance s^2 est non biaisé puisque :

$$
\begin{aligned}
E(s^2) &= E\left[\frac{1}{N-1}\sum_{i=1}^{N}(X_i-\overline{X})^2\right] \\
&= \frac{1}{N-1}E\left[\sum_{i=1}^{N}\big((X_i-\mu)+(\mu-\overline{X})\big)^2\right] \\
&= \frac{1}{N-1}E\left[\sum_{i=1}^{N}\big[(X_i-\mu)^2+2(X_i-\mu)(\mu-\overline{X})+(\mu-\overline{X})^2\big]\right] \\
&= \frac{1}{N-1}E\left[\sum_{i=1}^{N}(X_i-\mu)^2+\sum_{i=1}^{N}2(X_i-\mu)(\mu-\overline{X})+\sum_{i=1}^{N}(\mu-\overline{X})^2\right] \\
&= \frac{1}{N-1}E\left[\sum_{i=1}^{N}(X_i-\mu)^2+2(\mu-\overline{X})\sum_{i=1}^{N}(X_i-\mu)+n(\mu-\overline{X})^2\right] \\
&= \frac{1}{N-1}E\left[\sum_{i=1}^{N}(X_i-\mu)^2+2n(\mu-\overline{X})(\overline{X}-\mu)+n(\mu-\overline{X})^2\right] \\
&= \frac{1}{N-1}E\left[\sum_{i=1}^{N}(X_i-\mu)^2-2n(\overline{X}-\mu)^2+n(\mu-\overline{X})^2\right] \\
&= \frac{1}{N-1}E\left[\sum_{i=1}^{N}(X_i-\mu)^2-n(\overline{X}-\mu)^2\right] \\
&= \frac{1}{N-1}\left[\sum_{i=1}^{N}E((X_i-\mu)^2)-nE((\overline{X}-\mu)^2)\right] \\
&= \frac{1}{N-1}\left[\sigma^2-n\sigma^2/N\right] \\
&= \sigma^2
\end{aligned}
\tag{II.2}
$$

ANNEXE III

GÉNÉRATIONS DE NOMBRE ALÉATOIRE SUR GPU

Sami Hissoiny

École polytechnique de Montréal, Département de génie informatique et génie logiciel, 2500 chemin de Polytechnique. Montréal (Québec), CANADA H3T 1J4

Benoît Ozell

École polytechnique de Montréal, Département de génie informatique et génie logiciel, 2500 chemin de Polytechnique. Montréal (Québec), CANADA H3T 1J4

Philippe Després

Département de Physique, Université de Montréal, Pavillon Roger-Gaudry (D-428), 2900 Boulevard Édouard-Montpetit, Montréal, Québec H3T 1J4, Canada and Département de Radio-oncologie, Centre hospitalier de l'Université de Montréal (CHUM), 1560 Sherbrooke est, Montréal, Québec H2L 4M1, Canada

ABSTRACT

The future of high-performance computing is aligning itself towards the efficient use of highly parallel computing environments. One application where the use of massive parallelism comes instinctively is Monte Carlo simulations, where a large number of independent events have to be simulated. At the core of the Monte Carlo simulation lies the Random Number Generator (RNG). In this paper, the massively parallel implementation of a collection of pseudo-random number generators on a graphics processing unit (GPU) is presented. The results of the GPU implementation, in terms of samples/s, effective bandwidth and operations per second, are presented. A comparison with other implementations on different hardware platforms, in terms of samples/s, power efficiency and cost-benefit, is also presented. Random numbers generation throughput of up to ≈18MSamples/s have been achieved on the graphics hardware used. Efficient hardware utilization, in terms of operations per second, has reached ≈98% of the possible integer operation throughput.

III.1 Introduction

Over the years, the increase in computing power has been driven by the increase in processor clock speed. Lately, however, the increase in computing power has been achieved by the use of multi core CPUs (Central Processing Units) while raw clock frequencies have evolved at a slower pace. Recent trends in scientific computing point towards massively parallel computing devices to handle much of the workload. One class of hardware that has always been designed with parallelism in mind, due to the specialized task it accomplishes, is the Graphics Processing Unit (GPU). The scientific computing community is turning more and more towards this hardware platform as it is well suited for a class of problem often encountered when the intended goal is to simulate real phenomena: embarrassingly parallel computations. One family of applications among this paradigm, Monte Carlo simulations, rely on high-quality pseudo-random numbers. In this paper, we implement three algorithms for generating random numbers: the multiply-with-carry (MWC), the XorShift, and the KISS

(keep it simple stupid). These implementations of random number generators (RNG) will be analyzed in terms of their samples/s capability as well as their ability to exploit the resources of the GPU. It should be noted that the RNG of choice when long periods are wanted, the Mersenne Twister, has not been implemented for reasons that will be explained later.

Random number generators have been implemented on traditional CPUs as well as more parallel computing oriented platforms such as GPUs (Langdon, 2009; Janowczyk et al., 2008) and FPGAs (Sriram and Kearney, 2007; Thomas et al., 2009). A comparison to these implementations is also presented.

III.2 Background

III.2.1 Random number generators

RNGs in general have a history of bad implementations, as pointed out by the work of Park and Miller (Park and Miller, 1988). For this reason, well established random number generators will be used in this work, which is focused on their implementation on GPUs. This work is based on the RNGs by George Marsaglia (Marsaglia, 2003b; Marsaglia, 2003a). More precisely, a *multiply-with-carry* generator, a *XorShift* generator and the *KISS* generator have been implemented.

RNGs are based on the execution of mathematical operations on an ensemble of variables (the RNG state) to generate a sequence of numbers that will appear random; being generated through an algorithm they are obviously completely deterministic. The sequence takes the form of:

$$x, f(x), f^2(x), f^3(x), ... \tag{III.1}$$

where x is the first value, or set of values, supplied to the RNG, called the seed, and $f^2(x)$ is equal to $f(f(x))$. This recurrence will generate a seemingly random sequence until it has been called a number of times equal to its period. After this point, the sequence starts over.

III.2.1.1 multiply-with-carry

The multiply-with-carry (MWC) generator has been introduced by Marsaglia in 1991 (Marsaglia and Zaman, 1991). The generator takes the form of:

$$x_{n+1} = (a * x_n + c_n) \bmod (b) \tag{III.2}$$

where a is the multiplier and b the base. The carry, c, is defined by:

$$c_{n+1} = \lfloor \frac{(a * x_n + c_n)}{b} \rfloor. \tag{III.3}$$

When using integer arithmetic on computers, it is advantageous to use a base such as 2^{16} or 2^{32} in order to avoid the costly operations required to find the carry and replace them by very efficient shift operations. The carry of the operation $a * x$ with base b is $c = a * x >> b$ where $>>$ is the shift right (shr) operation. The modulo operation can also be transformed into an ADD operation. Indeed, $a \bmod (2^{16}) == a \&\& 2^{16}$,

The choice of the multiplier a is not arbitrary: the multiplier is chosen so that $ab - 1$ is safeprime. The period of the RNG with such a multiplier is on the order of $(ab - 1)/2$.

This generator is characterized by a very short state vector of only one element of w bits where w is usually 16 or 32. *Lagged* MWCs can also be developed. These *lagged* have a longer state vectors which act as a circular buffer of seeds for the MWC generator.

III.2.1.2 XorShift

The XorShift generator was introduced by Marsaglia (Marsaglia, 2003b) and further improved by Panneton (Panneton and L'Ecuyer, 2005) and Brent (Brent, 2004). The improvements proposed by Panneton will be used in this work. This RNG, as its name implies, is based on the successive application of xorshift operations on the state vector. The xorshift operation consists in replacing the w-bit block by a bitwise xor (exclusive or) of the original block with

a shifted copy of itself by a position either to the right or to the left, where 0 < a < w and w = 32 or 64. These generators have periods equal to $2^{rw} - 1$.

The generators are characterized by a number of xorshift operations on the state vector to output one random number. The mathematics behind this are well beyond the scope of this paper and are expertly explained by Panneton (Panneton and L'Ecuyer, 2005). Of particular interest is that the RNG is parametrized by r, the number of w-bit elements in the vector and s the number of xorshift operations. The sequence is defined by:

$$v_i = \sum_{j=1}^{r} \{A_j v_{i-j}\} \tag{III.4}$$

where the v_i's are w bits long and form the state vector and the A_j's are the product of v_j xorshift matrices. There are s non-zero A_j matrices.

The generators are very fast since they only require bitwise and shift operations on unsigned integers, which are generally some of the fastest instructions on a processor.

Panneton proposes a full period generator with $r = 8$, $s = 7$ resulting in a period of $2^{256} - 1$. The generator has the following recurrence:

$$\begin{aligned} v_n = (I + L^9)(I + L^{13})v_{n-1} + (I + L^7)v_{n-4} \\ +(I + R^3)v_{n-5} + (I + R^{10})v_{n-7} + (I + L^{24})(I + R^7)v_{n-8} \end{aligned} \tag{III.5}$$

where L is a left shift matrix and R a right shift matrix and the exponent the number of bits being shifted.

III.2.1.3 KISS

The KISS (*keep it simple stupid*) has been proposed by Marsaglia as the combination of three RNGs: multiply-with-carry, xorshift and congruential (Park and Miller, 1988). The

first two RNGs have already been discussed in previous sections. The XorShift generator is however different from the one that has already been presented. Indeed, the original XorShift generator proposed by Marsaglia will be used (Marsaglia, 2003b). The proposed generator is a type I XorShift generator (Panneton and L'Ecuyer, 2005). These generators have $r = 1$, meaning that the state is represented by only one $w = 32$ bits variable, and A_1 is the product of three xorshift matrices or, in other words, $s = 3$. The following A_1 matrix will be used, as proposed by Marsaglia:

$$A_1 = (I + L^{13})(I + R^{17})(I + L^5) \tag{III.6}$$

The above XorShift generator was shown by Panneton (Panneton and L'Ecuyer, 2005) to fail randomness tests of the TestU01 suite by L'Écuyer *et al.* (L'ecuyer and Simard, 2007). However, as will be explained later, combining several random number generators can have a positive effect on the randomness as well as the period of the resulting generator.

The new RNG introduced in this section as part of the combined RNG, the linear congruential generator (LCG) (Park and Miller, 1988), is similar to the MWC generator. The recurrence has the following form:

$$X_{n+1} = (aX_n + c)mod(m) \tag{III.7}$$

where a is a multiplier, c the increment (as opposed to the carry in the MWC) and m the modulus. Again, the choice of a is not left to chance. In order to obtain the full period $p = m - 1$, the modulo a must be prime and a must be a primitive element modulo m (L'ecuyer, 1999). However, since the objective of this work is the implementation of efficient RNGs, we will forgo the full period LCG and opt for an efficient modulo of 2^{32}, which is essentially free through integer overflow. Marsaglia proposes the values $a = 69069$, $c = 1234567$ and $m = 2^{32}$ (Marsaglia, 2003b).

The final random number value is obtained by xoring the MWC and LCG intermediate values to which the XorShift value is added.

III.2.1.4 Parallelizing RNGs

If RNGs are not suitably parallelized, running them on a parallel architecture could result in every substream outputing the same exact sequence of pseudo-random numbers, thus rendering useless the parallel architecture or its resulting correlated streams. For example, if only different seeds are sent to every node, the seed sent to node j could be the second random number generated by node i, making substreams i and j identical but lagged by one.

Two main categories of parallelization can be used for generating non-correlated streams on a parallel architecture: those that split a given generator into multiple substreams and those that generate multiple independent streams (Badal and Sempau, 2006). From the first category, we again find two subcategories: sequence splitting and leapfrogging.

In sequence splitting, a processor or node j will generate the random numbers $[R_j, R_{j+1}..R_{j+B}]$ where B is the amount of random numbers generated by each node and R_i is a random number. It can easily be seen that this can cause problems if it is not known beforehand how many numbers a given node will need to generate and if this amount happens to be larger than B. This technique can be employed with RNGs capable of computing the i'th random number generated without computing all the intermediate numbers. From this property, every node j can be seeded with the last value that would have been generated by node *j-1*. The generator itself does not have to be modified.

In leapfrogging, node j generates the random numbers $[R_j, R_{j+N}, R_{j+2N}, ...]$ where N is the number of processors executing the RNG. It is clear that in this case the sequences will not overlap. This technique also requires a way to compute the random number i without having to compute all the intermediate values. In this case, the RNG has to be modified in order to skip N numbers every time it has to generate one number.

Finally, the second category of RNG parallelization, called parameterizing, achieves uncorrelated sequences by effectively using a different RNG for all nodes. These different RNGs can be of the same type, *v.g.* a MWC generator, but with different generator parameters, *e.g.* with different multipliers m.

III.2.2 Graphics cards for scientific computing

The recent interest for graphics card computing within the scientific community stems from the fact that GPUs (Graphics Processing Units) offer a low cost alternative to traditional CPU clusters for parallel computations. Graphics hardware being inherently parallel, it is useful for a large array of scientific computations such as dense linear algebra (Garland et al., 2008), molecular dynamics (Anderson et al., 2008), N-Body simulations (Nyland et al., 2007), and so forth.

This work relies on the NVIDIA CUDA (NVI, 2009) platform which allows the programming of graphics cards using a syntax that is heavily based on the C language. The CUDA platform is available for all graphics card of the G80 family and beyond. The Unified Shader Model was introduced in this family, which effectively transformed the graphics card into a massively parallel SIMD (single instruction multiple data) or, as NVIDIA puts it, SIMT (single instruction multiple threads) machine where several hundred of identical computing elements, Scalar Processors (SPs) in NVIDIA terminology, are present.

The MultiProcessor (MP), a group of 8 SPs, is equipped with 16KB of shared memory, a fast cache that is available to all SPs of a MP. All the SPs also have access to the large global memory pool (256 MB and up). The global memory however presents bandwidth and latencies that are vastly inferior to shared memory. Current GPUs are equipped with 1 to 30 (or 60 in a dual-GPU graphics card) MPs for a total of up to 240 (or 480) SPs. Two Special Function Units (SFUs) are also present, capable of executing transcendental functions or a floating point multiply operation.

The global CUDA paradigm consists in launching thousands of threads, grouped in *thread blocks*. These blocks will be assigned to a given MP. A given MP can be assigned to multiple thread blocks that could be scheduled to launch concurrently (in warp locksteps) if the resources of the MP allow for more than one block to be launched, *v.g.* if there are enough shared memory and registers to accommodate both blocks.

The smallest computational group is called the *warp* and in the GT200 architecture is

composed of 32 threads. An SM will select a warp that is ready to execute and every thread of the warp will execute the same instruction.

Integer multiplications are considered slow on NVIDIA graphics card. Indeed, the integer multiplication of two 32 bits operands has a throughput of 2 operations per clock cycle whereas the throughput of the single precision floating point multiplication has a throughput of 8 operations per clock cycle. This could have an important impact on the performance of the MWC RNG. Other integer operations useful for the scope of this paper, such as bitwise XORs, shifts and additions have a throughput of 8 instructions per clock cycle. The GPU also provides a lower precision 24 bit integer multiplication with a 32 bits results and an 8 instructions per clock cycle throughput. However, NVIDIA announced that in future hardware versions, the lower precision multiplication will become slower than the full 32 bit multiplication.

III.3 Methods

III.3.1 GPU implementations

This section will cover the implementation details for the different RNGs that have been used. It is however necessary to explain why the Mersenne Twister (MT) (Matsumoto and Nishimura, 1998) has not been implemented, which is the RNG of choice when an extremely long period is required. The MT RNG requires a fairly large state vector (624 integers, 2496 bytes) in order to generate random numbers. This state has to be stored somewhere and access to fast enough memory is not sufficient. Consequently, the state vector would need to be stored in global memory on the GPU, therefore severely hindering performances. Indeed, since the thread blocks will usually contain more than 32 threads and that each thread needs its own state vector, there is not enough registers or shared memory space to store the state vectors of a thread block. It has therefore been decided to use RNGs with relatively small state vectors at the cost of a reduced period. The NVIDIA CUDA SDK already has an implementation of the MT19937 random number generator which will be included in the

comparison to our results. NVIDIA has used global memory to store the large state vector.

III.3.1.1 multiply-with-carry

In the current implementation, two MWCs are combined together as has been originally proposed by Marsaglia (Marsaglia, 1997). There is evidence (L'ecuyer, 1988) that combining two or more RNGs not only increases the period but also the randomness of the sequence. These two MWCs therefore need independent multipliers. Since the 32-bit arithmetic and logical units (ALU) of the GPU are used, the multiplier a and carry c values have to fit inside 16 bits and the base b is set to 2^{16}. There is a limited number of choices (392 to be exact) for the multiplier a that satisfy the conditions exposed in III.2.1.1. This is seemingly not enough for the thousands of independent RNGs being launched, in accordance with the CUDA paradigm. However, it is the combined results of two MWCs to generate one sequence of random number per thread that is of interest. It is therefore the number of distinct combination of two multipliers that needs to be sufficient, and that number (k-combination of 2 in 392) is plentiful.

The MWC generator is therefore parallelized through parametrization where independent streams are generated by each RNG (or thread). A different set of seeds (two) is also given to every thread. Each thread generates a number of random numbers (>1000) before exiting.

The execution flow for the CUDA kernel starts by having each thread fetch its corresponding seeds and multipliers values from global memory. These memory transactions are completely coalesced. A loop having as many iterations as the numbers of random numbers to be generated by every thread then begins. The core of the loop generates two intermediate numbers, one for every intermediate MWC. These two 16 bits intermediate values are concatenated to form the final 32 bits integer random number. This number is written to global memory.

The total number of useful operations for the core of the loop to generate one random number is 10 (3 $shifts$, 2 $ands$, 2 $mults$ and 3 $adds$). This is confirmed by inspection of the generated assembly code, with the addition of address calculations for the storage of the random number

and loop overhead.

Fourteen registers are used by this implementation. Using the NVIDIA occupancy calculator, the optimal block sizes of 128, 256 or 512 are found, which give an occupancy of 1.0 on the NVIDIA GTX280 card.

III.3.1.2 XorShift

The XorShift generator used in this work is the one proposed by Panneton (Panneton and L'Ecuyer, 2005) and presented in Section III.2.1.2.

Using values of $r = 8$ and $w = 32$, a state vector of eight 32 bit values has to be stored for every generator or, in other words, every thread. Ideally, this state vector is to be stored in registers. However, as of CUDA 2.3, the registers are not indexable, *v.g.* the [] operator cannot be used, which is required by the algorithm (Panneton and L'Ecuyer, 2005) to address the state vector. Shared memory, which is addressable, has therefore been used, as a first approach to store the state vector. Each thread then needs $r*w = 256$ bits of shared memory.

To fully exploit the bandwidth of the on-chip shared memory, certain guidelines have to be followed (NVI, 2009). Shared memory is divided into banks that can be accessed simultaneously. If N requests fall into N different memory banks, these requests can then be serviced simultaneously. In the GT200 architecture used for this work, shared memory is divided into 16 banks (equal to the number of threads in a half-warp) and successive 32 bits words are in successive banks. It is therefore important to allocate the vector in shared memory in a way that will not cause bank conflicts, where two requests would be sent to the same memory bank. The scheme (explain the scheme shortly) presented in Figure III.1, which assigns successive state vector element for a given thread with a *blockDim* stride (where *blockDim* is the size of the CUDA thread block), has been used resulting in a bank conflict free implementation.

The rest of the XorShift algorithm maps natively to the programming model in CUDA as

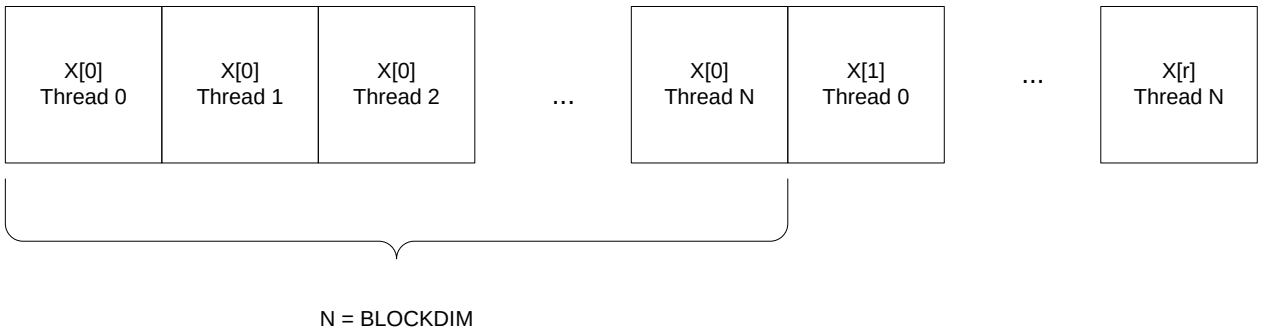

Figure III.1 Shared memory arrangement for the XorShift RNG.

only $shift$ and XOR operations are needed. The execution flow for the CUDA kernel starts with each thread fetching its correspondent 8 seeds from global memory and storing them to the shared memory. Again, these memory transactions are completely coalesced. A loop having as many iterations as the number of random numbers to be generated by every thread then begins. The core of the loop operates on two intermediate values t and y. These t and y values are modified through the recurrence presented in Section III.2.1.2. The resulting y value is written back to the state vector in shared memory as well as to the results array in global memory.

The number of useful operations to generate one random number is 18 (11 $XORs$ and 7 $shifts$). The C implementation proposed by Panneton (Panneton and L'Ecuyer, 2005) outputs one random number every time it is called. It therefore needs to implement an implicit circular buffer over the r elements of the state vector and increment the current vector element pointer, k, every time the routine is called. Consequently, the address and index calculations must be determined at runtime for every generated random number since the indices depend on k. Logical *and* operations also have to be used to implement a circular buffer over the r elements. This adds many operations that are not directly useful to the generation of random numbers. It results that 52 assembly operations are required to generate one random number.

We have no doubt that Panneton (Panneton and L'Ecuyer, 2005) proposed the implementa-

tion as an "inline" random generator where one random number is requested for every call. It has been implemented as is as an experiment. However, since the goal of this work is to generate a large quantity of random numbers, we can implement the circular buffer explicitly with r consecutive statements in a manner similar to loop unrolling. With this technique, r random numbers are generated for every execution of the main loop. The total number of assembly instructions is brought down to 23 per random number.

At this point, there is no need for an indexable memory space since the circular buffer has been made explicit. It is therefore possible to replace the shared memory array of r elements with r registers. This decision should be made by taking into consideration which resource (registers or shared memory) will be most likely limited when the RNG is sampled "on-the-fly" by a complete simulation engine also requiring resources. By using registers instead of shared memory, the total number of assembly instructions per random number is found to be an optimal 18.

The number of registers used by this implementation is 14 for the explicit loop and 15 with the unrolled implementation. Using the NVIDIA occupancy calculator, the optimal block sizes of 128, 256 or 512 are found, which give an occupancy of 1.0 on the NVIDIA GTX280 card.

III.3.1.3 KISS

The implementation of the KISS generator relies on the previously discussed implementation of the MWC generator, together with a simplified XorShift generator and the congruential generator.

The simplified XorShift generator has $r = 1$, $w = 32$ and only one unsigned integer is therefore needed to store the state vector. The use of shared memory is consequently avoided. The newly introduced LCG RNG is implemented with a single line of code that can be easily derived from the recurrence presented in Section III.2.1.3.

The same parallelization schemes have been used for the two previously discussed RNGs: different multipliers as well as different seeds for the MWC generator and different seeds to the XorShift generator. The LCG is parametrized using the multiple seeds scheme as well.

The state vector for the combined generator is therefore composed of four 32 bits words (2 for the MWC, 1 for the simplified XorShift and 1 for the LCG). In addition to these seeds, the two multipliers of the MWC also have to be fetched from global memory. The rest of the implementation is similar to what has previously been mentioned in the two previous sections.

The number of useful operations for the main loop of combined generator is found to be 18. In terms of useful operations it is therefore equivalent to the XorShift RNG. However, the need for shared memory has been completely avoided which translates in a reduction of the number of overhead instructions associated with the memory address calculation when compared to the implicit implementation of the XorShift generator. This is apparent in the assembly code file where fewer lines of code are generated for the KISS RNG compared to the implicit XorShift RNG. The newly introduced LCG RNG uses a 32 bit integer multiplication which has a throughput four times lower than the other integer operations used.

III.3.2 Performance evaluation

For all implementations, approximately 80 millions random numbers are generated by each call to the random number generator. The random numbers are written to global memory. They could therefore be used by subsequent CUDA kernels or copied back to the computer's main memory for use by the CPU. The execution times also include the writing of the state vector to global memory for subsequent use. The timing results do not include the copying of the results back to the system's main memory. The rate at which the random numbers are generated, in samples per second, will be reported.

Furthermore, the rate at which the random numbers are generated when there is no need to write them back to global memory has also been studied. This result is of interest for

applications that would inline the random number generator instead of generating a pool of random numbers for later use. To achieve this result, intermediate results are not written to global memory until the end of the main loop, at which point the state of the RNG is written to memory to ensure that the whole loop is not simply suppressed by optimization during compilation.

The achieved memory bandwidth (BW) and usable operations per seconds will also be presented with regards to the theoretical limit of the hardware used. The memory bandwidth is measured as:

$$BW = \frac{n_t * N_t * 4}{t * 10^9} GBps \tag{III.8}$$

where n_t is the number of random numbers generated per thread, N_t the number of threads invoked, t the execution time and the 4 factor is used to account for the size $w = 32bits = 4bytes$ of the generated number. It will only be of interest when the results are written back to global memory. The number of usable operations per second (U_{ops}) result is measured as:

$$U_{ops} = \frac{n_t * N_t * n_{ops}}{t * 10^9} GOps \tag{III.9}$$

where n_{ops} is the number of operations required to generated a random number as presented in Section III.3.1. The simpler FLOPs (floating point operation per second) measure cannot be used here since no floating point operations are taking place except when the generated number is scaled back to [0,1[range.

Two types of random numbers generators have also been implemented, resulting in two types of random numbers: unsigned integers in the [0,2^{32}-1] range and single-precision floating-point numbers in the [0,1[range. The latter requires two extra steps to convert the integer number to a floating point representation and to scale it to the [0,1[range through a single-precision floating-point multiplication.

The main loop, which generates a series of random numbers within a kernel, has been unrolled using the supplied *pragma* to the *nvcc* compiler in order to reduce the loop control logic overhead. The compiler has chosen not to unroll the main loop of the XorShift generator

with the explicit circular buffer most likely due to register pressure.

Finally, the randomness of the implemented RNGs has been assessed with the use of the DIEHARD (Marsaglia, 1995) and TestU01 (L'ecuyer and Simard, 2007) applications. TestU01 supplies three levels of randomness testing: *SmallCrush, Crush* and *BigCrush*. The *Crush* battery of tests requires approximately 2^{35} random numbers to complete. The DIEHARD battery of test is less stringent, as is apparent by the relative time it takes to complete when compared to TestU01: around 10 seconds compared to one hour. The quality of the three random number generators has been studied, first by the use of the DIEHARD battery of test, then by the *Crush* battery of tests from TestU01.

The results have been obtained with an NVIDIA GTX280 graphics card with 1GB of global memory and on a Xeon Q6600 processor. The CUDA *kernels* have been compiled with the *nvcc* compiler included in the version 2.3 of the toolkit and the CPU programs with the -O2 options activated using the compiler included with Microsoft Visual Studio 2005.

III.4 Results

In the following tables, the results are presented for the unsigned integer random generator. The first column indicates if the results are written back to memory. The last column of the tables, $uniform\ rate$, presents the rate for a uniformly distributed random number generator.

Tables III.1, III.2 and III.3 present the results for the GPU implementation of the MWC, XorShift and KISS RNGs, respectively.

Table III.1 Results for the MWC generator.

	t (ms)	BW $GBps$	U_{ops} $GOps$	$rate$ $GSamples/s$	$uniform\ rate$ $GSamples/s$
W/ writeback	2.3	67.2	168.1	18.0	17.9
W/o writeback	1.9	-	300.6	37.5	23.6

Table III.2 Results for the XorShift generator.

		t (ms)	BW $GBps$	U_{ops} $GOps$	$rate$ $GSamples/s$	uniform rate $GSamples/s$
Implicit	W/ writeback	9.1	16.6	74.9	4.4	4.3
	W/o writeback	8.5	-	81.1	4.8	4.4
Explicit	W/ writeback	5.1	30.2	145.7	8.1	7.6
	W/o writeback	3.0	-	248.2	13.8	11.5
Registers	W/ writeback	3.3	46.4	224.1	12.5	11.5
	W/o writeback	2.8	-	264.5	14.5	12.9

Table III.3 Results for the KISS generator.

	t (ms)	BW $GBps$	U_{ops} $GOps$	$rate$ $GSamples/s$	uniform rate $GSamples/s$
W/ writeback	4.4	34.2	153.9	9.2	8.5
W/o writeback	2.6	-	304.8	19.1	10.3

III.4.1 Randomness

The randomness of our implementation of these RNGs has been studied with DIEHARD and the *Crush* battery of tests from TestU01. All random generators have passed all tests from the DIEHARD battery of tests. The KISS generator has passed all tests from the *Crush* battery of tests while both the XorShift and MWC have failed exactly 1 test, out of the 96 applied, from the *Crush* battery of tests.

III.4.2 Comparison with other studies

Table III.4 presents a comparison between our GPU implementations, the equivalent single threaded CPU implementation as well as a collection of random number generators found in the literature. The comparisons to other GPU work has to take into consideration the fact that different GPUs have been used to generate the results. Thomas *et al.* (Thomas et al., 2009) have used an NVIDIA GTX280, Langdon (Langdon, 2009) a NVIDIA 8800GTX and

Janowczyk *et. al* (Janowczyk et al., 2008) a GeForce 8800GTS. It should also be noted that the XorShift implementation presented by Thomas *et al.* is not equivalent to ours: it uses $r = 5$ and the number of $xorshift$ operations, s, is unknown. Their implementation thus uses less memory for the state vector than ours as well as possibly requiring less instructions to generate one random number.

A study of the efficiency of the implementations, in terms of MSamples/Joule, has been performed as well as a cost-benefit study in terms of MSamples/s/$. In order to acquire the power consumption of every devices in our comparison, the TDP (thermal design power) value has been used. An online retailer's prices have been used with the exception of the GeForce 8800GTX and 8800GTS which are no longer available for sale.

Table III.4 Comparative results.

	Efficiency 10^6 Samples/Joule	Cost-Benefit 10^6 Samples/s/$	Throughput 10^9 Samples/s
MWC	76.3	78.3	17.9
XorShift	52.9	54.3	12.5
KISS	36.0	36.7	8.5
MWC_{CPU}	2.8	1.4	0.3
$XorShift_{CPU}$	1.0	0.5	0.1
$KISS_{CPU}$	1.9	0.91	0.2
$MT19937_{FPGA}$ [1] (Sriram and Kearney, 2007)	-	-	119.6
$SFMT_{CPU}$ [2]	32.7	13.6	4.3
$XorShift_{GPU}$	71.5	73.4	16.9
$Lut-Opt_{FPGA}$ (Thomas et al., 2009)	8636.7	108.2	259.1
$Park-Miller_{GPU}$ (Langdon, 2009)	5.4	-	≈ 1
$LaggedFibonacci_{GPU}$ (Janowczyk et al., 2008)	10.7	-	1.3
$MersenneTwister_{GPU}$ (Podlozhnyuk, 2007)	12.7	13.0	3.0

III.5 Discussion

Tables III.1, III.2 and III.3 provide an overview of the different RNGs implemented in this work. The GTX280 has a theoretical FLOPs value of 933. The figure assumes that each SP is capable of executing 3 FLOPs per cycle (one fused multiply-add on the SP and one multiply on the SFU). Very special circumstances have to be met to sustain the theoretical FLOPs figure. In the present case, no fused multiply-add will take place and no operation will be sent to the SFU since only integer math is used. The maximal achievable operations per second figure in our case is therefore one third of the maximum theoretical FLOPs figure. To this inherent limitation, memory calculation arithmetic as well as loop overhead have to be acknowledged when looking at the achieved operations per second figure. The theoretical memory bandwidth of the NVIDIA GTX280 is 141.7 GBps. Our own benchmark has revealed that the achievable bandwidth is closer to 117GBps.

The U_{ops} value for the two RNGs not requiring shared memory, the MWC and the KISS generators, show good agreement with the above discussion. They respectively achieve 96.7% and 97.8% of the maximum theoretical operations per second.

In the XorShift RNG with the implicit circular buffer, the overhead of index and address calculations become apparent. Indeed, 6 instructions are required to fetch one piece of data from shared memory. The overhead would be the same if registers were indexable so it is not a question of shared memory versus indexable registers. In the end, more than half of the operations are required by the index calculation and implementation of the circular buffer, and thus not directly used for random number generation, which explains the low U_{ops} value of 26%. With the explicit version of the same RNG yielding the same results, the observed U_{ops} value raises to 79% of the maximum theoretical operations per second. When avoiding shared memory completely and using only registers to store the state vector, this figure raises to 85%.

It can also be seen that the cost of converting the integer value to the [0,1[range is almost completely hidden by the bandwidth limitation as there is almost no difference between the

integer and uniform samples/s rate when the results are written back to memory. However, when the RNGs are used "on-the-fly", requiring no writes to global memory, the conversion and subsequent normalization are responsible for a performance drop of almost 50% in the worst case.

The achieved memory bandwidth does not go over 50% of the observed achievable bandwidth. Our explanation is that this behavior is related to the fact that only memory writes take place during the main loop of the *kernels*. The same results are obtained when modifying the benchmark used to find the achievable bandwidth so that only memory writes occur.

III.5.1 Comparison with other studies

It is clear that GPUs are an attractive platform for the generation of random numbers, as this study and other studies have shown.

The proposed MWC generator is the fastest GPU-based RNG found in the literature. Its implementation falls natively within the GPUs strength, using only the fastest operations available on the GPU will requiring a small vector state. The proposed XorShift generated is slower than the XorShift generator proposed by Thomas *et al.* but, as it was said, we are unsure of the actual number of operations required by their implementation to generate one random number, which directly impacts the *rate* value. We have followed Panneton's recommendation of using more than three XorShifts operations on the state vector to achieve acceptable randomness and we are again unsure if their implementation respects this.

The FPGA platform offers a seemingly unparalleled solution to random number generation. One aspect to keep in mind is that the FPGA implementations presented here most likely fully use the configurable hardware present on the FPGA to generate the reported number of samples per second. In a real world situation, the generated random numbers have to serve a purpose. For example, they could be used within a Monte Carlo simulation. The FPGA RNG implementations would therefore have to allocate a significant portion of the configurable hardware to the simulation logic, reducing their random number production rate accordingly.

The generated numbers could also be transferred through one of the FPGA's extension card to some other hardware platform responsible for the simulation, consequently being limited by the connection between the two devices and once again sacrificing the effective production rate. The higher cost of FPGAs also explains a much smaller gap in the cost-benefit column of Table III.4 between FPGAs and GPUs.

III.6 Conclusion

This paper has presented the GPU implementation and parallelization of three random number generators: the multiply-with-carry, the XorShift and the KISS. For each generator, the exploitation of the GPU's resources have been evaluated. The results indicate that the GPU is an attractive platform for generating random numbers. It has also been shown that the intensive use of shared memory as an indexable storage space comes with a high address calculation cost and a significant loss of useful operations per second performance.

III.7 Acknowledgements

This work has been supported by the Natural Sciences and Engineering Research Council of Canada CRSNG/NSERC.